Universidad Zaragoza
1542

Trabajo Fin de Máster

Diseño y construcción de una placa electrónica para el control de una bobina Tesla

Design and building of an electronic board for a control of a Tesla coil

Autor

Julio Sebastián Sullca Trillo

Directores

Francisco José Pérez Cebolla
Jesús Letosa Fleta

Máster en Ingeniería Industrial

Escuela de Ingeniería y Arquitectura
2021

RESUMEN

Las bobinas de Tesla de las que dispone la EINA cumplen una función didáctica. Son utilizadas en demostraciones científicas para instituciones educativas. Su particular funcionamiento despierta el interés de los más jóvenes sirviéndoles como incitación para continuar con una formación científica.

Actualmente estas carecen de un sistema de control automático para regular el funcionamiento a través de la frecuencia de excitación. Poder controlar este parámetro nos permitirá desde estudiar el comportamiento de la carga en el dominio de la frecuencia, hasta encontrar el punto de consumo de potencia óptimo.

La bobina de Tesla es un circuito con dos frecuencias de resonancia. El comportamiento singular aparece en el entorno de éstas. Alternándose entre inductivo, resistivo y capacitivo en un ancho de banda muy estrecho donde las corrientes y tensiones tienen altas derivadas respecto de la frecuencia, además de producir un fuerte ruido electromagnético a su alrededor, lo que puede producir problemas en los circuitos electrónicos cercanos. La finalidad de este trabajo fin de master es dotar a la universidad de una placa electrónica robusta para la oscilación forzada, capaz de soportar las condiciones de trabajo descritas. Para lograrlo se diseñará y fabricará un puente inversor de onda completa.

Empezaremos el trabajo con un estudio en Matlab de la respuesta en frecuencia. Con la finalidad de definir los rangos de funcionamiento y el dimensionamiento de componentes. Una vez analizada la respuesta de este tipo de carga, diseñaremos y construiremos el inversor de puente en H. El puente diseñado deberá ser robusto frente a los picos de corriente y potencia. Inmune al ruido electromagnético, además de poseer protecciones para evitar condiciones límites de temperaturas y corrientes. Asimismo, se programará el firmware necesario para controlar el inversor. El control aplicado deberá ser externo a la placa, guardando una distancia segura entre la electrónica de control y la alta tensión. Con ese fin, la comunicación entre el controlador y la etapa de potencia se realizará por fibra óptica. Siendo esta inmune al ruido electromagnético. Finalizaremos el proyecto con los ensayos y análisis de resultados sobre una bobina de Tesla real.

El trabajo aporta una solución para el control en frecuencia de una bobina de Tesla. Asimismo, posibilitará abordar posteriores estudios experimentales relacionados con este tipo de bobinas u otros sistemas compatibles con las características del puente inversor construido.

Índice

1. Introducción

La bobina de Tesla fue construida por Nikola Tesla a finales del siglo XIX. Es un transformador resonante, con núcleo de aire que alcanza altas tensiones en su circuito secundario, capaces de ionizar el aire y producir descargas eléctricas [1].

El objetivo de Tesla era utilizar esta tecnología para enviar electricidad a largas distancias de forma inalámbrica, utilizando para ello un sistema de varias torres distribuidas en distintas ubicaciones, todas ellas en resonancia, con un funcionamiento similar al de una antena de radio. Con la diferencia de transmitir elevadas potencias [2].

Finalmente, la idea resultó irrealizable. Actualmente la bobina se usa con fines didácticos en laboratorios. No obstante, la transmisión de energía inalámbrica es posible a cortas distancias. Hoy en día podemos encontrar cargadores inalámbricos en distintos rangos de potencia. Desde cargadores para ordenadores o móviles hasta los desarrollados para la carga rápida de vehículos eléctricos.

En este trabajo fin de master se ha diseñado y construido un sistema electrónico de potencia para alimentar una de las bobinas de Tesla disponibles en el Departamento de Ingeniería Eléctrica de la Universidad de Zaragoza. La diferencia entre esta versión y otros diseños previos es que dispondrá de protecciones contra sobretensiones, sobrecorrientes, sobretemperaturas. De igual modo con el fin de proteger el sistema de control de la distorsión del campo electromagnético provocado por el funcionamiento de la propia bobina, la comunicación entre este y la etapa de potencia se ha realizado por fibra óptica, estando a su vez dicha etapa embebida en una caja apantallada. Además, mediante la modificación del sistema de control previo de tipo analógico a uno digital y programable, se ha posibilitado que este sea capaz de establecer de forma automática la frecuencia óptima de operación del puente, salvando posibles tolerancias o desajustes en la construcción de la bobina e incluso el uso de otras bobinas con el mismo puente. Por último, dado su carácter didáctico, el sistema se ha construido siguiendo un diseño modular orientado a la intercambiabilidad de bloques operativos según la funcionalidad deseada.

1.1 Estado del arte

Los transformadores resonantes no han cumplido el objetivo que Nikola Tesla concibió en el siglo XIX. Sin embargo, no han caído en desuso. Gracias al gran avance de la electrónica hoy en día se pueden fabricar mecanismos de carga inalámbrica basados en la tecnología resonante. Siendo una ventaja donde el cableado sea un inconveniente, como es en el caso de los móviles y los vehículos eléctricos [3] [4]. Su principal ventaja frente a los transformadores convencionales es que al carecer de núcleo no tienen perdidas en la transferencia de energía de un devanado a otro. Además, la resonancia permite la carga ultrarrápida. Siendo esta otra gran ventaja, ya que hoy en día la velocidad de carga es uno de los grandes inconvenientes de las baterías. La carga inalámbrica se está convirtiendo en un factor importante para el desarrollo de los coches eléctricos [5].

En lo que respecta a la Escuela de Ingeniería y Arquitectura de Zaragoza (EINA) podemos afirmar que están familiarizados con la utilización de las bobinas de Tesla. Diversos proyectos de años anteriores lo avalan (parte de este trabajo utiliza como punto de partida [6] y [7]). Tenemos a nuestra disposición tres bobinas de Tesla, con diferentes características. Dos de ellas han sido adquiridas a proveedores comerciales, mientras que la tercera ha sido diseñada y construida por estudiantes de cursos anteriores. Estas bobinas se muestran en la ilustración 1.



Ilustración 1 Bobinas de Tesla del departamento. *La bobina de la derecha se corresponde con un proyecto anterior. Las otras dos son modelos comerciales.*

En cuanto a la excitación de la bobina, hay 2 tipos de tecnologías que se pueden utilizar. La primera de la que vamos a hablar es por oscilación natural. Este es el modelo clásico y se basa en la frecuencia de resonancia propia de un conjunto LC. Consiste en dejar cargar este circuito para después hacerlo oscilar a su frecuencia natural. Actualmente disponemos de este mecanismo de excitación. Igual que el diseño original mediante explosor.

La otra opción para el funcionamiento es por oscilación forzada. La diferencia con el diseño original es que podremos imponer la frecuencia de excitación de la bobina. Anteriormente la Escuela disponía de un prototipo de oscilación forzada [6]. Este diseño nos servirá como punto de partida. Actualmente la placa, que podemos ver en la ilustración 2, se encuentra inoperativa. La oscilación forzada se conseguía utilizando un inversor de puente completo. El objetivo de nuestro proyecto será la construcción de un segundo prototipo.

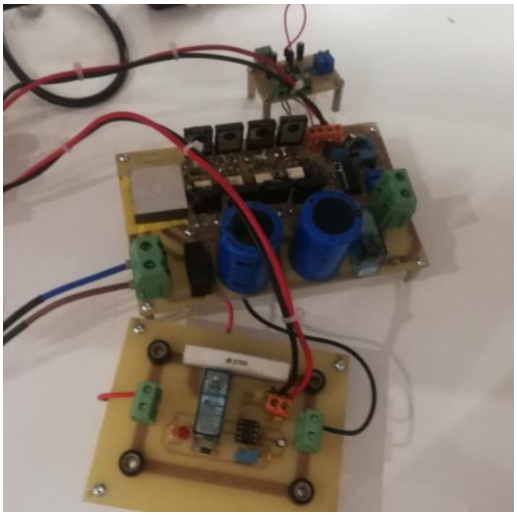


Ilustración 2 Primer prototipo de inversor de puente completo. *La selección de la frecuencia y otros parámetros de funcionamiento se realizaba mediante un ajuste manual.*

1.2 Definición, motivación y alcance

La finalidad de este proyecto es la construcción de un inversor en puente en H, con el reto de que la electrónica sea capaz de controlar de forma segura potencias cercanas a 1kW a frecuencias de hasta 200kHz. La placa será diseñada para soportar el comportamiento particular que representa la bobina de Tesla. Aunque podrá utilizarse con cualquier otra finalidad compatible con dicho inversor.

El proyecto surge como idea de mejorar el prototipo de oscilación forzada del que se disponía anteriormente en el laboratorio. Haciendo una segunda versión más robusta y con más prestaciones que la original.

Se diseñará y construirá un inversor de puente completo. El funcionamiento deseado se conseguirá actuando sobre la señal de control de los IGBTs. Esta será controlada externamente mediante un microprocesador o un ordenador. En este caso se utilizará un Arduino Due.

El Arduino Due enviará la señal de paso a ON y paso a OFF de los IGBTs a través de un cable de fibra óptica. Evitando así cualquier posible interferencia electromagnética. A su vez el microcontrolador recibirá una realimentación de la placa con los parámetros necesarios para llevar a cabo un control automático (valores de tensión, corriente, potencia y alertas de seguridad). Se utilizará una comunicación serie entre la placa de potencia y el microcontrolador mediante una UART.

Nuestro diseño está orientado a proteger los transistores de potencia y a la fácil reposición de los elementos más susceptibles de deterioro. Para ello se tomarán medidas como el aislamiento galvánico de componentes, diferenciación de los circuitos de potencia, control y alimentación. Además de protecciones asíncronas contra sobrecorrientes, sobretemperaturas, etc.

Con el objetivo de reducir el deterioro de la placa por los altos campos eléctricos producidos durante el funcionamiento de la bobina. Se diseñará un sistema de apantallamiento eléctrico para la placa. Otra característica nueva es la utilización de la fibra óptica para transmitir las señales de control. La utilización de fibra óptica aumenta la inmunidad al ruido eléctrico. En la ilustración 3 podemos ver el diagrama de bloques del sistema.

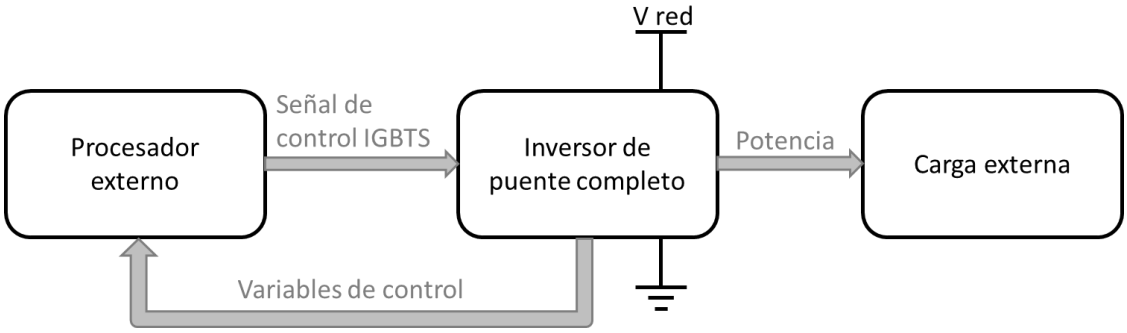


Ilustración 3 Esquema de funcionamiento. *Controlaremos los IGBTs modificando la frecuencia, el dead time entre ramas del inversor y las ráfagas de pulsos que dejamos pasar de la señal de control.*

2. Modelo eléctrico de la bobina de Tesla

Como paso previo al diseño del sistema, se debe conocer el funcionamiento de la carga a controlar. En la ilustración 4 se puede observar el esquema del circuito de excitación original de una bobina de Tesla.

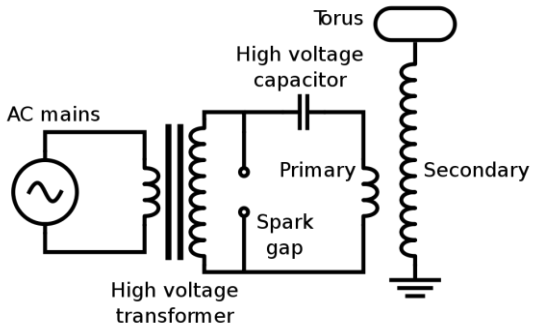


***Ilustración 4 Diseño original de una bobina de Tesla** [1].*

En el diseño concebido por Nikola Tesla. La alimentación de la carga se realizaba con un transformador elevador conectado a una fuente de corriente alterna a frecuencia de red. La carga que alimenta este trasformador es otro transformador elevador. En este caso resonante y con núcleo de aire (la bobina de Tesla). Formado por un circuito LC en el devanado primario y otro en el devanado secundario (se han obviado las componentes resistivas). El circuito LC de ambos devanados deben tener la misma frecuencia de resonancia.

El funcionamiento está basado en que a bajas frecuencias el comportamiento del sistema es capacitivo. Viene marcado por el condensador del primario. Este se irá cargando hasta alcanzar un valor de tensión lo suficientemente alto para hacer saltar una descarga en el explosor. En este momento el circuito se cierra y el explosor actuará como un cortocircuito. Cuando el explosor se activa el sistema oscila de forma natural. El condensador del primario se volverá a cargar una vez que la tensión no sea la suficientemente alta para hacer saltar un arco eléctrico en el explosor. Este es el funcionamiento por oscilación natural.

Sin embargo, el avance de la electrónica ha permitido otro diseño conocido como bobina de Tesla de estado sólido. El funcionamiento es por oscilación forzada. Consiste en alimentar el circuito del primario de la bobina con un generador electrónico de alta frecuencia, sintonizado a la frecuencia de resonancia o a la que deseemos trabajar. La gran diferencia es que en este diseño podemos imponer la frecuencia a la que el sistema oscila. Lo que nos permitirá controlar parámetros como la potencia que se le entrega. En la ilustración 5 podemos observar un esquema del circuito por oscilación forzada.

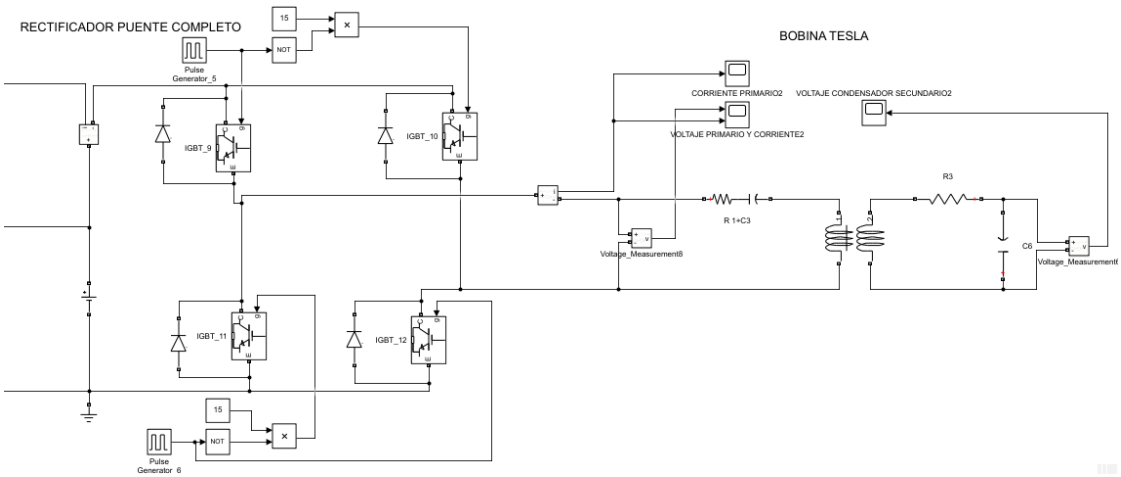


***Ilustración 5 Diseño por oscilación forzada**. Esquema de Simulink*

En el diseño original de Tesla el funcionamiento era fijado por los elementos del sistema y por tanto no controlable. El diseño por oscilación forzada es más versátil. Ahora podremos modificar libremente la frecuencia a la que alimentamos la bobina de Tesla. El hecho de poder controlar esta nos permitirá también controlar el comportamiento del sistema. Destacamos que, al utilizar un inversor en puente en H, la salida del mismo es una onda cuadrada de tensión.

2.1 Estudio analítico

En este apartado se estudiará la respuesta de una bobina de Tesla. Para simplificar los cálculos se obvian los elementos resistivos del devanado. El acoplamiento de las dos bobinas puede expresarse con el circuito equivalente de la ilustración 6. Con la peculiaridad de que si bien formalmente el modelo es correcto, este no es un circuito físicamente realizable. Esto es debido a que el valor de M puede llegar a ser mayor que L1, dando lugar a inductancias negativas.

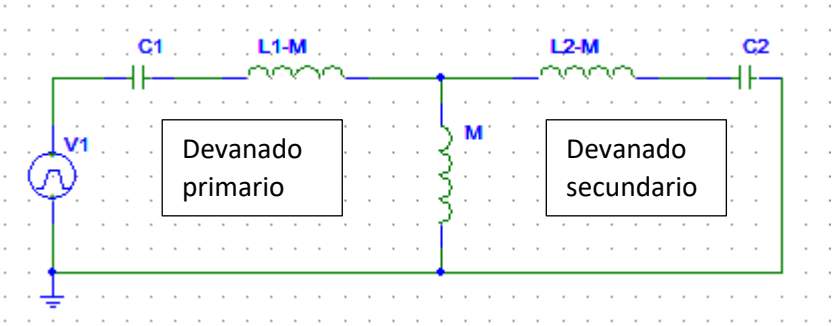


Ilustración 6 Circuito equivalente de una bobina Tesla. *C1 y L1 son el condensador y la bobina del devanado primario y C2 Y L2 los correspondientes al devanado secundario. M es el coeficiente de inducción mutua.*

La resonancia de un circuito LC se alcanza cuando la impedancia capacitiva e inductiva se compensan entre sí. Podemos extraer la frecuencia de resonancia de las ecuaciones 1 y 2.

$$\frac{1}{C\omega} = L\omega \tag{1}$$

$$f_{resonancia} = \frac{1}{2\pi\sqrt{CL}} \tag{2}$$

Tenemos un circuito LC en cada devanado. Así, para que ambos circuitos tengan la misma frecuencia natural se deberá cumplir la condición indicada en la ecuación 3.

$$L_1 \cdot C_1 = L_2 \cdot C_2 \tag{3}$$

Sin embargo, debido al acoplamiento de los devanados tendremos 2 frecuencias de resonancia distintas. El valor de estas vendrá determinado por las ecuaciones 4 y 5.

$$f1 = \frac{1}{2\pi} \cdot \sqrt{\frac{1}{LC(1+K)}} \tag{4}$$

$$f2 = \frac{1}{2\pi} \cdot \sqrt{\frac{1}{LC(1-K)}} \tag{5}$$

Donde K es el coeficiente de acoplamiento entre los dos circuitos. Mientras que LC es el producto de la capacidad del condensador por la inductancia de la bobina del primario o del secundario. Idealmente la relación entre los dos es la misma.

Si se observa de nuevo la ilustración 6, es fácil deducir que la resonancia tendrá lugar cuando la impedancia que forman el conjunto serie del devanado secundario en paralelo con la inductancia M se compense con la impedancia del conjunto serie del devanado primario. La existencia de dos frecuencias de resonancia se analiza haciendo uso de la transformada de Laplace [8] y cuyo desarrollo se muestra en el anexo de cálculos. Así, la función de transferencia entre el voltaje que se aplica y la corriente que se aporta a la carga o impedancia equivalente posee 4 polos y 2 ceros. Por cada 2 polos, siendo estos complejos conjugados, hay una frecuencia de resonancia. Lo mismo ocurre con los ceros.

El cero del sistema es un punto donde la corriente será nula y la impedancia infinita. Esto sucede cuando la impedancia en paralelo entre el devanado secundario y la inductancia M es máxima. En la ecuación 6 podemos ver la impedancia del paralelo correspondiente.

$$\frac{M\omega j\cdot((L2-M)\cdot\omega j-\frac{j}{C2\omega})}{M\omega j+(L2-M)\cdot\omega j-\frac{j}{C2\omega}} \tag{6}$$

A la frecuencia natural del devanado secundario el denominador se anula. Por lo tanto, a esta frecuencia la impedancia tiende a infinito. Esta será la frecuencia de consumo mínimo ya que la corriente por el circuito se hace cero. Como conclusión extraeremos las frecuencias características de una bobina de Tesla en base a las ecuaciones 7, 8, 9 y 10.

$$\omega_{natural} = \frac{1}{\sqrt{C1\cdot L1}} = \frac{1}{\sqrt{C2\cdot L2}} \tag{7}$$

$$\omega_{resonancia\ 1} = \frac{\omega\ natural}{\sqrt{1+k}} \tag{8}$$

$$\omega_{resonancia\ 2} = \frac{\omega\ natural}{\sqrt{1-k}} \tag{9}$$

$$\omega_{consumo\ mínimo} = \frac{1}{\sqrt{C2\cdot L2}} \tag{10}$$

Al no haber considerado las componentes resistivas, matemáticamente la corriente será infinita a la frecuencia de resonancia y cero a la frecuencia de mínimo consumo. No obstante, si bien, la influencia de las resistencias en la determinación de las frecuencias es mínima estas limitarán el comportamiento del sistema. Es decir, la corriente no podrá ni ser infinita ni tampoco nula, quedando acotada entre dos valores. Mientras que, en la aplicación real los valores máximos de corriente y potencia estarán definidos por las propias limitaciones constructivas de los componentes. En la ilustración 7 podemos observar el resultado de una simulación realizada en Matlab con el objeto de facilitar la compresión de los expuesto anteriormente. En esta se muestra la variación del módulo de la impedancia equivalente de una bobina con la frecuencia.

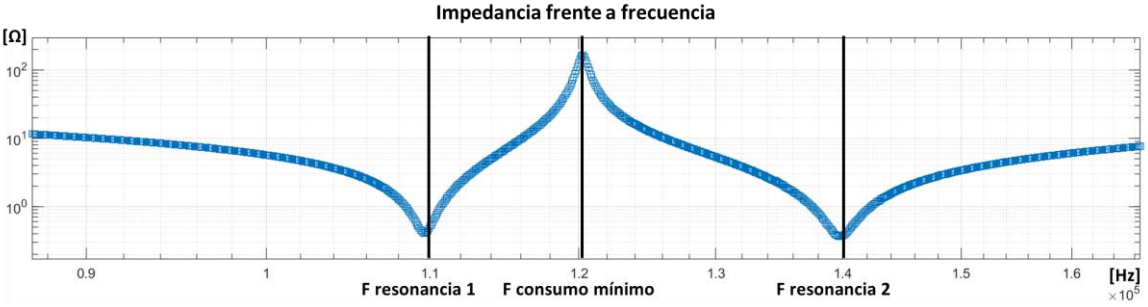


Ilustración 7 Módulo de impedancia equivalente frente a la frecuencia.

2.2 Análisis del circuito con Matlab

En este apartado se simulará el circuito equivalente de una bobina de Tesla de parámetros conocidos haciendo uso del software Matlab. Esta será la utilizada al final del trabajo en las pruebas experimentales. Con este estudio buscamos entender el comportamiento que tiene la bobina respecto de la frecuencia. Lo que nos permitirá determinar parámetros de diseño y rangos de funcionamiento sobre los que dimensionar nuestra placa. También conoceremos los valores de tensión, corriente y potencia que deberíamos esperar durante la prueba final. La primera parte del estudio analizará la respuesta de la bobina en frecuencia utilizando los diagramas de Bode. En este primer estudio se considerará un comportamiento de régimen permanente. El código utilizado aparece en el anexo de firmware.

En la segunda parte se construirá un simulador del puente inversor y carga en Simulink. El cuál nos permitirá observar el comportamiento real de la bobina de Tesla teniendo en cuenta tanto el régimen transitorio como permanente a una frecuencia específica.

2.2.1 Respuesta en frecuencia

En este apartado analizaremos como la frecuencia de excitación modifica algunas de las variables más importantes del sistema. Analizaremos con el diagrama de bode la impedancia equivalente, la corriente total consumida, el voltaje del condensador secundario y las potencias aparentes, reactivas y activas. El diagrama de Bode también nos servirá para determinar numéricamente las frecuencias de resonancia. La bobina viene caracterizada por los valores de la tabla 1.

Designación	Valores	Definición
R1	284,74 mΩ	Resistencia del devanado primario
L1	22,015 µH	Inductancia del devanado primario
C1	75 nF	Condensador del devanado primario
M	0,32 mH	Coeficiente de inducción mutua
R2	352 Ω	Resistencia del devanado secundario
L2	83,4 mH	Inductancia del devanado secundario
C2	21 pF	Condensador del devanado secundario
K	0,23	Coeficiente de acoplamiento
Frec. 1	107566 Hz	Frecuencia de resonancia inferior teórica
Frec. 2	138866 Hz	Frecuencia de resonancia superior teórica
Frec. valle.	120262 Hz	Frecuencia de mínimo consumo teórica

Tabla 1 parámetros eléctricos de la bobina simulada

A partir de la simulación se obtendrán los valores de tensiones, corrientes y potencias cuando alimentamos la bobina con una onda cuadrada. La señal de salida (onda cuadrada) se obtiene a partir de la red mediante un puente rectificador de diodos y un inversor enlazados en tensión mediante un condensador de filtrado. La amplitud de la señal cuadrada debería ser aproximadamente 325V. No obstante, el valor de esa tensión será menor debido al rizado del condensador, quedando los valores calculados ligeramente por encima de los consumos reales sobre los que diseñar nuestra placa. Para simplificar los cálculos se ha aproximado la señal

cuadrada por su primer armónico. Los coeficientes de la serie de Fourier de una onda cuadrada son los que vemos en la ilustración 8.

$$a_n = 0$$

$$b_n = \frac{1}{\pi}\left(\int_{-\pi}^{0} -\sin nt\,dt + \int_{0}^{\pi} \sin nt\,dt\right) = \begin{cases} \dfrac{4}{\pi n}, \text{ si } n \text{ es impar;} \\ 0, \text{ si } n \text{ es par.} \end{cases}$$

Ilustración 8 Coeficientes de la serie de Fourier de una señal cuadrada. *Una señal cuadrada solo tiene coeficientes de Fourier impares [9]*

La amplitud del primer armónico será aproximadamente 413V, resultado de multiplicar la amplitud de la onda cuadrada por un factor de 4/π. Los valores de tensión y corriente que representemos serán la amplitud de la onda. Para las potencias se ha utilizado el valor de tensión eficaz del primer armónico 292V. Destacamos que excepto en la gráfica de impedancia equivalente las magnitudes que veremos son proporcionales a la tensión de entrada. Una vez definidos los parámetros que modelan nuestro circuito y la alimentación podemos empezar a simular la respuesta en frecuencia.

Impedancia equivalente

La ilustración 9 se corresponde con el diagrama de Bode de la impedancia equivalente del sistema.

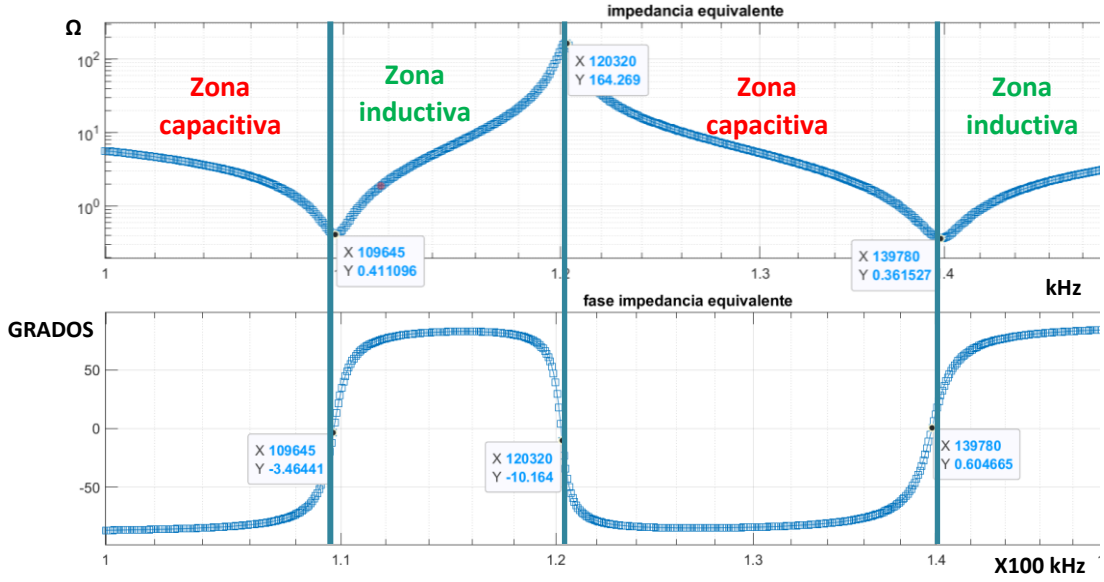


Ilustración 9 Gráfica Z-f

Podemos observar claramente cuáles son las frecuencias características de la impedancia. Los valles son las resonancias y el pico es el cero del sistema. Los 3 puntos tienen comportamiento resistivo ya que en ellos las componentes imaginarias de la impedancia se compensan. Las frecuencias de resonancia donde la impedancia es mínima son 109,65kHz y 139,78kHz respectivamente. Mientras que la frecuencia de consumo mínimo, donde la impedancia es máxima, es 120,32kHz. Como podemos observar estos valores no difieren mucho respecto a los valores teóricos que aparecen en la tabla 1. Destacamos como varía el comportamiento de la carga según la frecuencia. Fuera de los límites de la gráfica, el

comportamiento se vuelve muy capacitivo a bajas frecuencias y muy inductivo a altas. Conforme nos acercamos a las frecuencias características el comportamiento alterna entre inductivo y capacitivo.

De esta gráfica extraemos los rangos de frecuencias óptimos para trabajar. Siempre que sea posible trabajaremos en la zona inductiva. Para que la conmutación sea del tipo ZVS. Evitándonos así las pérdidas de paso a ON en los transistores. Hablaremos de las perdidas en conmutación del inversor más adelante.

Corriente consumida

En la ilustración 10 podemos observar la corriente que se entrega a la carga respecto a la frecuencia de excitación. Veremos un rango cercano a la frecuencia de mínimo consumo.

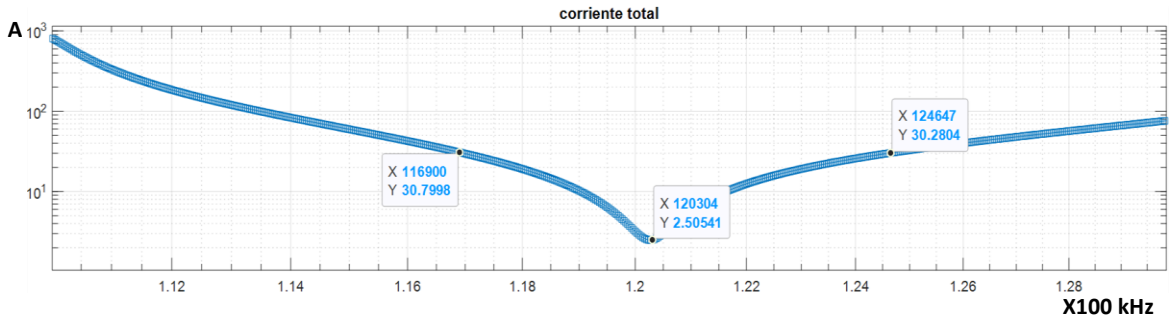


Ilustración 10 Gráfica I-F

La frecuencia de mínimo consumo se refiere al consumo mínimo de corriente. Sin embrago, si nos acercamos lo suficiente a las frecuencias de resonancia, la corriente alcanzará valores muy altos que podrían dañar la placa. En la ilustración 10 se observa que a 117kHz y a 124,5kHz la corriente está en torno a 30A mientras que en el valle la corriente es de 2,5A. Siempre que sea posible debemos trabajar en la frecuencia de corriente mínima.

La tensión máxima a la que se va a alimentar la placa es la de red, sin perjuicio de operar con tensiones menores en las pruebas experimentales con la finalidad de ampliar el rango de frecuencias de operación sin dañar los componentes. El valle será el punto óptimo de partida. Puesto que tiene consumos de corriente soportables y está en medio de las dos resonancias. Partiendo de él podremos desplazarnos a las frecuencias cercanas para evaluar la respuesta. A priori solo conocemos el valor teórico de este punto. Para encontrar el punto exacto debemos construir un inversor con una tolerancia en frecuencia capaz de soportar sin dañarse las condiciones de funcionamiento cercanas a la frecuencia de mínimo consumo. Para que después mediante un control automatizado el propio inversor sea capaz de sintonizarse a la frecuencia de corriente mínima.

Fijaremos la corriente máxima que deben soportar los componentes en torno a los 30 A. Será recomendable que para encontrar esta frecuencia se haga un ajuste a menor tensión. Aumentando así la tolerancia de frecuencias que la placa es capaz de soportar. A tensión de red el rango de operación irá de 117kHz a 124,5kHz. Esto nos deja un ancho de banda de 7,5kHz. Este será el rango de operación que veremos en el resto de las gráficas.

Voltaje de los condensadores

En la lustración 11 se observa la tensión del condensador secundario. Este es el voltaje que puede llegar a alcanzar la corona de la bobina a la tensión de red en el rango de frecuencias al que podemos trabajar.

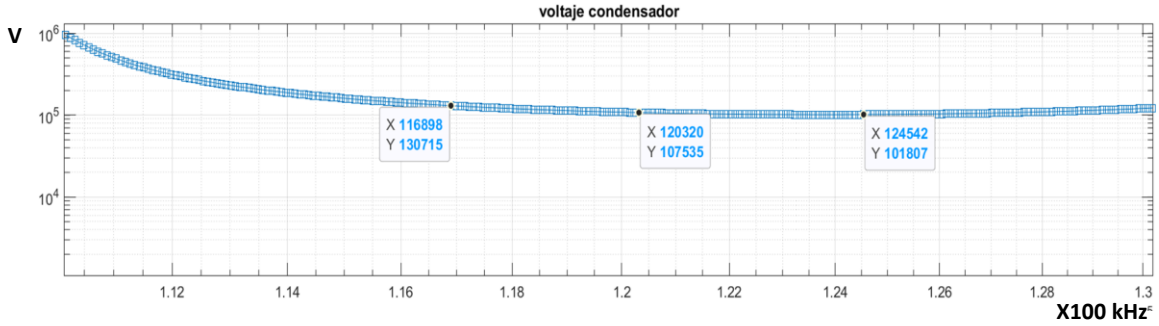


Ilustración 11 Voltaje del condensador secundario. *Rango 117-124 kHz*

Bajo estas condiciones de simulación la tensión en el condensador secundario permanecerá siempre por encima de los 100kV. Por otro lado, en la ilustración 12 veremos la tensión del condensador primario.

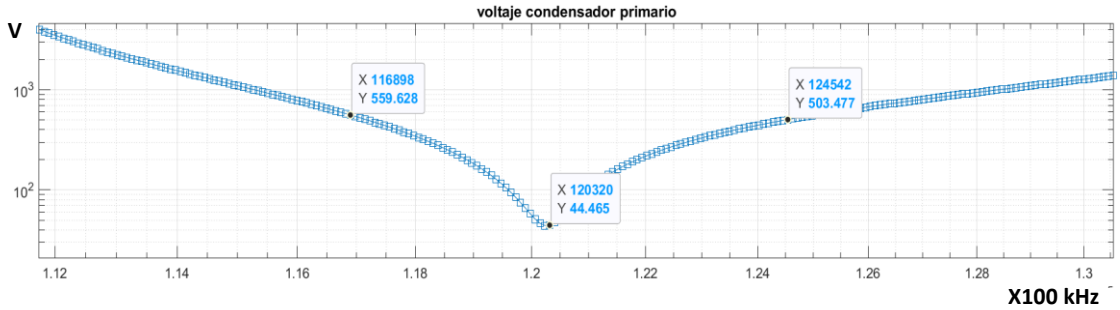


Ilustración 12 Voltaje del condensador primario. *Rango 117-124 kHz*

De esta gráfica destacamos que en el rango de frecuencias que hemos definido no se superaran los 2000V que tienen como límite los condensadores que utilizamos.

Potencias consumidas

Conocida nuestra frecuencia de partida (120,32kHz), vamos a evaluar cuál será el consumo de potencia en el rango de frecuencias que hemos definido cuando alimentamos a la tensión de red (ver ilustración 13).

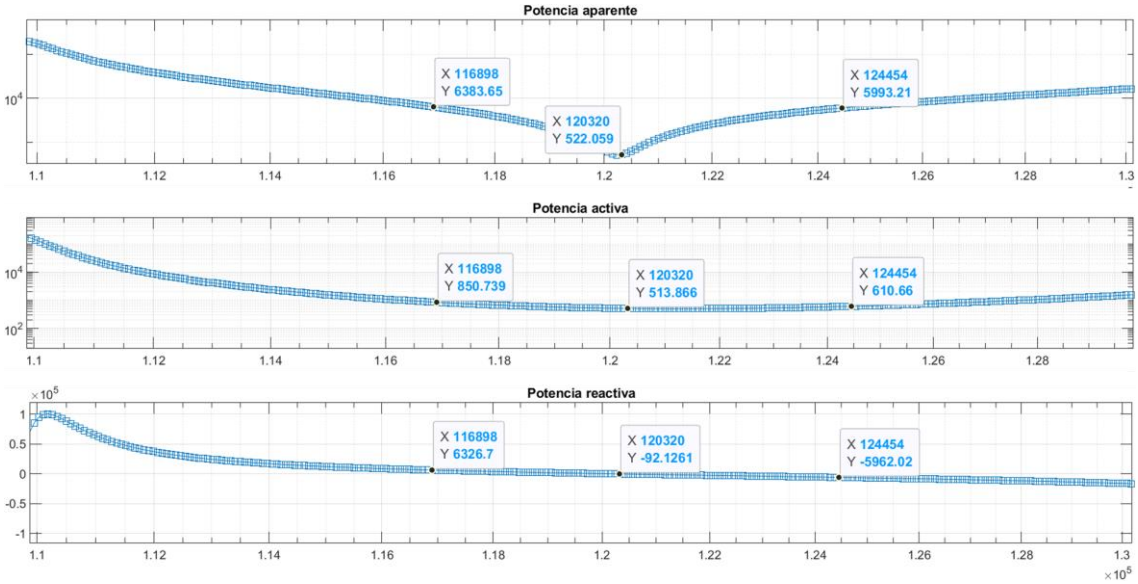


Ilustración 13 Potencias consumidas. *La reactiva es positiva si la carga es inductiva. Si es capacitiva la reactiva será negativa.*

La potencia a la corriente mínima es de 514W. Sin embargo, el punto de consumo mínimo de potencia estará ligeramente desplazado a la derecha. La diferencia de consumo entre ambos puntos es de 10W. En el rango de frecuencias que hemos establecido la potencia aparente alcanza valores del orden de 6kVA mientras que la potencia activa estará en torno a los 800W. Son valores muy altos teniendo en cuenta que aún no hemos modelado la descarga de la corona. Deberemos integrar un control de potencia para ampliar nuestra frecuencia de operación sin sobrepasar los límites constructivos de la placa. Consistirá en un control por ráfagas de los pulsos que van a la puerta de los transistores de potencia. Se dejarán pasar un porcentaje de pulsos de cada 100. De esta forma fraccionaremos la potencia que se entrega.

Conclusiones del estudio previo

- El punto de trabajo óptimo será el de mínimo consumo de corriente, equivalente a la frecuencia natural del devanado secundario. 120,32kHz para esta bobina.
- La máxima de corriente que vamos a soportar estará en torno de los 30A.
- Para reducir la potencia consumida fraccionaremos los pulsos de control dejando pasar una cantidad de pulsos definida de cada 100.
- La placa que diseñemos estará dimensionada para valores de potencia activa cercanos a 1kW.
- El rango de operación a tensión de red y máxima potencia estará entre 116,9kHz y 124,547kHz. Para ampliar este rango deberemos reducir la tensión de alimentación además de fraccionar la potencia entregada.

El análisis que hemos realizado hasta ahora nos da una estimación bastante cercana de cómo se va a comportar la bobina de Tesla en régimen permanente senoidal. Por otro lado, debemos destacar que la descarga del condensador secundario hacia el ambiente irá descargando progresivamente el sistema. Esto provocará que no alcancemos un régimen senoidal permanente. El comportamiento que veremos será el del régimen transitorio. Estudiaremos este comportamiento en Simulink. En la ilustración 14 podemos observar las partes de la respuesta en frecuencia donde se realiza cada estudio.

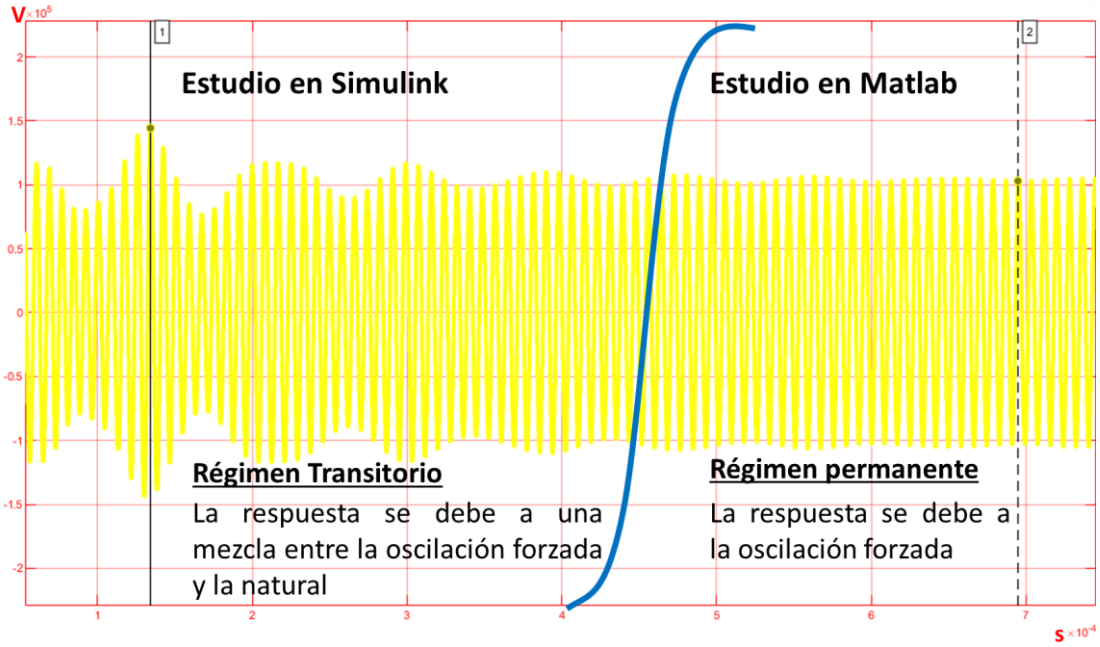


Ilustración 14 Zonas donde realizamos los estudios. *La señal amarilla es el voltaje del condensador secundario a la frecuencia de consumo mínimo*

2.2.2 Respuesta transitoria

Vamos a simular el comportamiento real en régimen transitorio. Para ello usaremos la herramienta de Matlab, Simulink. El modelo de Simulink es similar al de la ilustración 5.

Las condiciones de la simulación son las siguientes:

- Se usará una fuente ideal DC de potencia infinita de 325 V.
- Los IGBT del puente inversor funcionan como interruptores perfectos.
- La frecuencia del inversor es 120,32kHz.
- La descarga se modela con una resistencia variable, cuando el valor de tensión del condensador es máximo su valor será de 10Ω. El resto del tiempo esta será lo suficientemente alta para funcionar como un circuito abierto.

Podemos ver en la ilustración 15 la simulación de tensión en el condensador secundario. Se ha dejado que este se descargue por completo cuando alcanza su valor de tensión máxima.

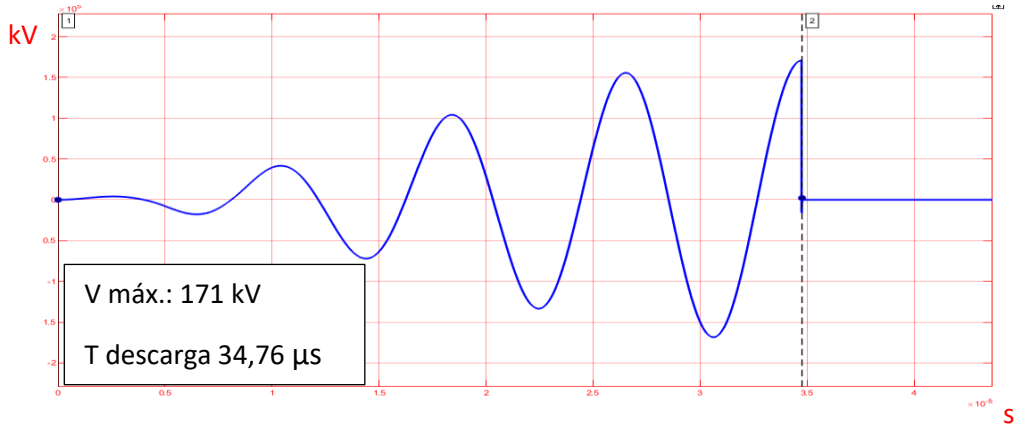


Ilustración 15 Tensión del condensador a 120,32kHz

En el régimen transitorio la tensión del condensador secundario tiene valores mucho más altos y muy oscilantes por lo que si se produce arco eléctrico este se producirá antes de alcanzar el régimen permanente, lo que descargará la corona de la bobina reiniciando el ciclo otra vez. La tensión del condensador secundario no alcanzará el régimen permanente debido a las descargas sobre el entorno. La potencia disipada al ambiente a traves del arco eléctrico que se forma en el condensador secundario y que provoca su descarga con una frecuencia de excitación de 120,32kHz viene determinada por la ecuación 11.

$$P = \frac{C\,V^2}{2} f_d = \frac{21x10^{-12}x171000^2}{2} x28785 = 8{,}73\ [kW] \quad (11)$$

Donde C es la capacidad del condensador de secundario respecto a tierra, V es la tensión de pico en la ilustración 15 y f_d la frecuencia de descarga. Es igual a la inversa del tiempo transcurrido hasta alcanzar la descarga que aparece en la ilustración 15.

A pesar de ser una aproximación bastante mayorada. Esta potencia esta muy por encima de los límites de operación de nuestra placa. Por lo que queda demostrada la necesidad de implantar un control en potencia. El cual nos debe permitir fraccionar esta.

3. Diseño y construcción de la placa

Nuestro punto de partida es el diseño de la placa de potencia anterior [6]. Podemos basarnos en los elementos que se utilizaron y en los cálculos que se realizaron. Por ejemplo, el IGBT elegido, la capacidad del condensador, el controlador analógico, etc.

La prioridad para la selección de componentes será su disponibilidad y que ya hayan sido probados. Las novedades de nuestro diseño serán la comunicación por fibra óptica, la implementación de un módulo para el sensado de datos, las protecciones y la inclusión del sistema digital de control auto sintonizable. Nuestro diseño deberá ser más robusto frente a mayores consumos de potencia y deberá tener unos rangos de funcionamiento definidos.

Al tener de referencia la placa anterior tendremos conocimientos de los puntos fuertes y débiles del modelo, lo que nos ayudará en la construcción del nuevo diseño. Se abordaron varias propuestas para el diseño de la PCB antes de dar con el planteamiento definitivo. Desde diseños en dos plantas con etapas diferenciadas a diseños más compactos de solo una planta, pero con funciones limitadas. Esto lo podemos observar en la ilustración 16.

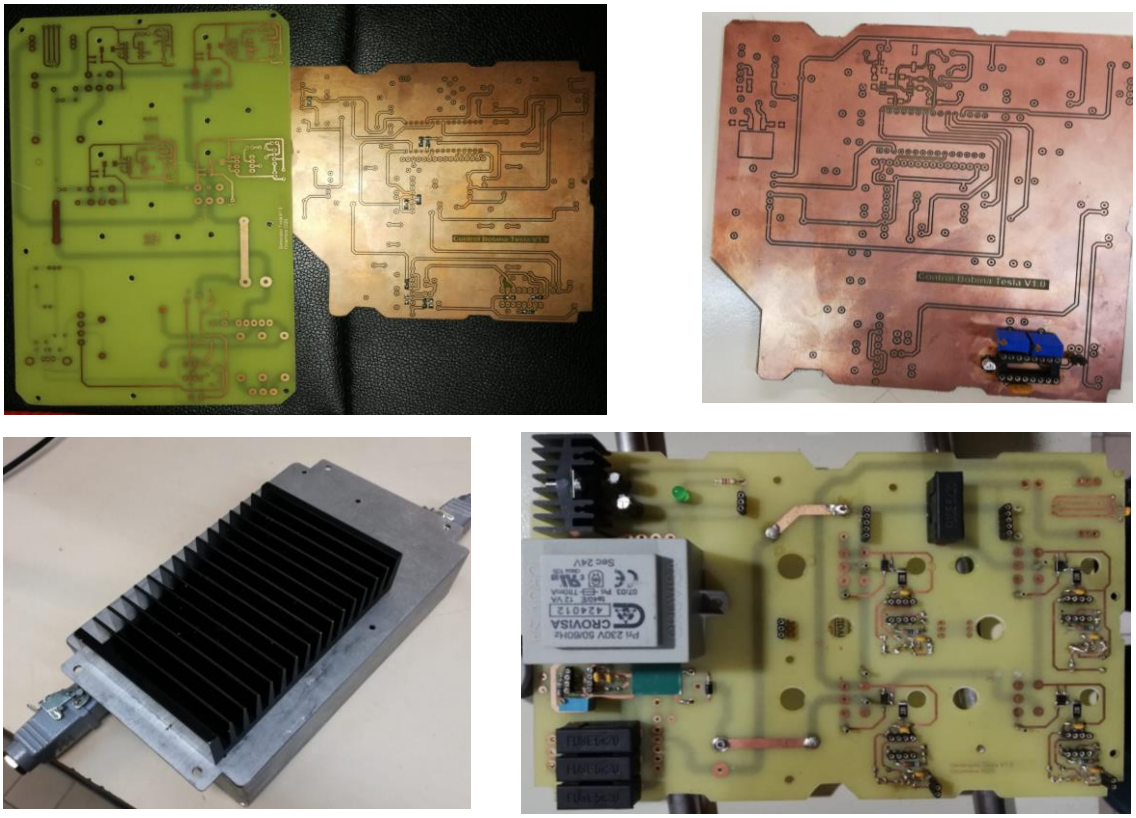


Ilustración 16 Primeros diseños. *La foto superior izquierda y derecha se corresponden con un diseño de dos plantas. Una para la electrónica de potencia y otra para la electrónica digital o de baja potencia. La foto inferior izquierda es una caja de apantallamiento para alojar la placa. La foto inferior derecha es un diseño compacto de una sola planta.*

Finalmente, para evitar los problemas que surgieron al principio y debido al carácter de aplicación docente se optó por realizar un diseño modular. Así, se ha fabricado una PCB por cada módulo según su funcionalidad. Por ejemplo, un módulo de IGBT, un módulo de fibra, etc. Estas PCBs se podrán conectar y desconectar de la placa principal mediante zócalos. Los fundamentos en los que nos apoyamos para el diseño que hemos implantado son:

- La intercambiabilidad de piezas. Si un componente falla solo habrá que sustituir su módulo por uno nuevo.
- Si un componente falla debemos evitar que este fallo se propague.
- Diseño más versátil. Si hay que añadir algún componente o funcionalidad nueva solo habrá que rediseñar el módulo correspondiente.
- Diseño por partes. Nos podremos centrar en el diseño de una sola parte operativa que será integrada al conjunto de la placa, en lugar de tener que pensar en un diseño de integración general.
- Diseño más amplio, para evitar problemas de espacio se optó por hacer la placa del tamaño que se necesite. El blindaje se realizará a posteriori.

Respecto a la parte de protecciones. En la placa principal se incluye un Arduino Nano con la finalidad de sensar los valores de corriente, voltaje, temperatura, alertas de sobrecorrientes y de sobretemperaturas. Valores que servirán de realimentación para un procesador externo.

Para el control del funcionamiento del puente actuaremos sobre la señal de disparo de los IGBTs. Esta se puede controlar mediante un microprocesador o un módulo generador de pulsos analógico. Se enviará la tensión de control a los drivers encargados de los IGBT mediante una

conexión de fibra óptica. Por fines didácticos, se ha respetado la posibilidad de generar internamente (sin fibra) los pulsos de disparo en la propia placa mediante un regulador analógico, lo cual posibilita el control manual de la frecuencia de operación mediante un simple potenciómetro (siendo hasta ahora este el único modo que había para controlar la placa). La comunicación entre el microprocesador y los IGBTs se realiza mediante 2 trenes de impulsos que circulan por 2 canales. Uno por cada rama del inversor. Las consignas de control que podemos modificar son la frecuencia de los pulsos, el dead time entre ramas y la fracción de pulsos que dejamos pasar de cada 100. En la ilustración 17 podemos observar un diagrama que ilustra la interconexión de las partes operativas en cada caso.

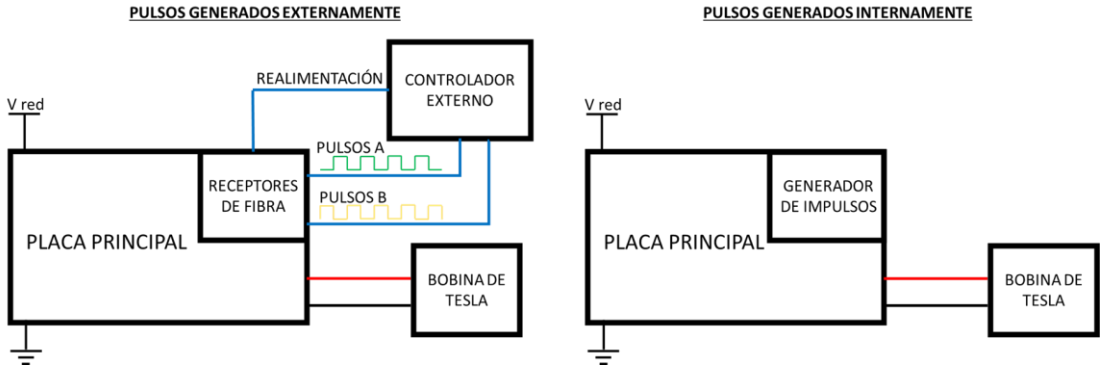


Ilustración 17 Conexiones entre módulo de control, placa principal y carga

3.1 Especificaciones de diseño

En este apartado se definirán las especificaciones que nuestra placa debe cumplir, así como los componentes que utilizaremos.

3.1.1 Potencia activa

El puente rectificador de diodos que se usará es trifásico (FUO50-16N) capaz de soportar hasta 50A de salida [10]. Este rectificador nos posibilita la conexión monofásica o trifásica. En este último caso la potencia entregada dependerá de la que sea capaz de dar la red dentro de los límites constructivos de la placa. Destacamos que en el laboratorio la tensión monofásica y trifásica es de 230V.

En las simulaciones se consideraba que teníamos una fuente de potencia infinita. En la realidad se puede considerar que nuestra fuente será el condensador del puente inversor. La energía que entregaremos vendrá limitada por la que el condensador sea capaz de almacenar manteniendo el nivel de tensión del bus sin muchas variaciones hasta el siguiente ciclo de carga. Siendo el rizado de la tensión proporcional a la potencia activa consumida. Como veremos a continuación hemos diseñado el inversor para que funciones con un condensador de 470µF. La tensión máxima que soporta el condensador instalado es de 400V.

Analizando el circuito rectificador. La corriente de descarga del condensador tiene la forma de la ecuación 12.

$$I = C\frac{dV}{dt} = C\frac{\Delta V}{T} = C\Delta V 2 f_{red} \rightarrow \frac{Vdc}{R} = C\Delta V f_{bus} \qquad (12)$$

Donde V_{dc} es el voltaje continuo promedio, mientras que $f_{bus.}$ es el doble de la frecuencia de red en el caso de la monofásica. Para la trifásica la frecuencia será 6 veces la de red [11].

$$P = \frac{Vdc^2}{R} \quad Riz = \frac{\Delta V}{Vpico} \quad Vdc = Vpico - \frac{\Delta V}{2} \tag{13}$$

Donde V_{pico} es la amplitud de la tensión 325V. De 12 y 13 podemos deducir la ecuación 14.

$$P = C\ Vpico\ Riz\ f_{bus}\ \left(Vpico - \frac{Vpico\ Riz}{2}\right) \tag{14}$$

Lo único que cambia en la configuración trifásica es la frecuencia. Con el condensador elegido y un rizado máximo definido al 15% obtendremos las potencias reflejadas en la tabla 2.

Vpico	325V	Potencia monofásica	**700W**
C (400V)	475µF		
Rizado máximo	15%		
f monofásica	100Hz	Potencia trifásica	**2100W**
f trifásica	300Hz		

Tabla 2 Potencias consumidas y valores de alimentación

La mayor ventaja de la trifásica es que podría funcionar perfectamente sin condensador. En este caso el rizado disminuirá la tensión hasta un valor mínimo de 282V. El valor de tensión medio será 310V y el de tensión eficaz es de 312V [11]. Volviendo al condensador, su carga ocurrirá 2 veces por ciclo de red en la configuración monofásica y 6 veces en el caso de la trifásica tal como se observa en la ilustración 18. Para un 15 % de rizado, cada pulso de recarga durará aproximadamente 2ms. La corriente máxima de carga será 25,28A en ambas configuraciones. El puente de diodos es capaz de soportar esta corriente sin problemas. Con estos parámetros la corriente eficaz en el caso de la monofásica será de 6,26A mientras que para la trifásica será de 10,85A. Debido a que esta tiene más ciclos de recarga. Sabiendo que la configuración predeterminada será monofásica podemos elegir unos los fusibles de protección de un valor de 10A. En el anexo cálculos podemos ver el desarrollo para la estimación de las corrientes.

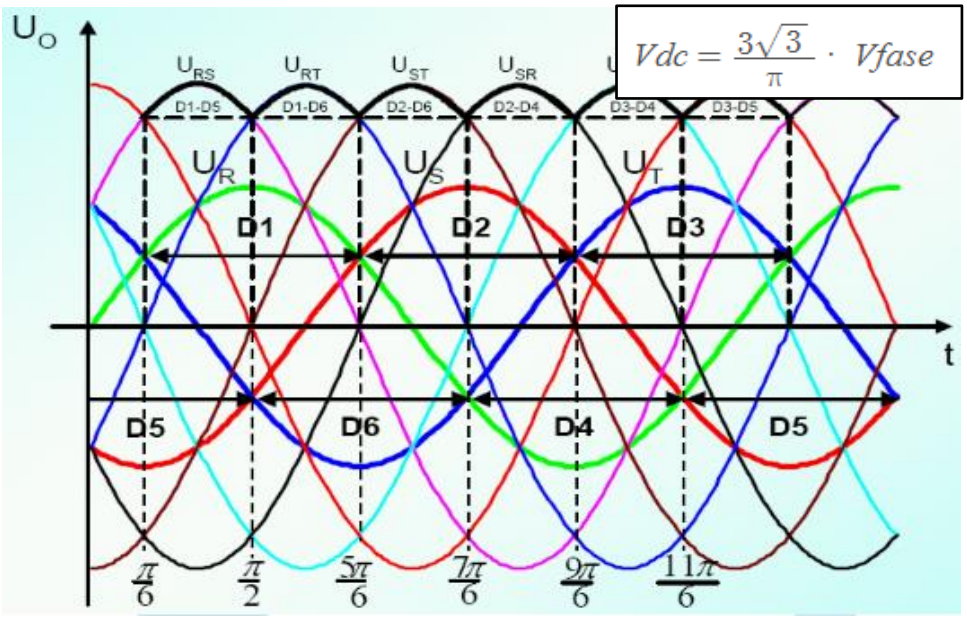


Ilustración 18 Onda de salida del puente rectificador trifásico. *[10]*

3.1.2 Puente inversor de onda completa

El corazón del inversor son los transistores que utiliza. La primera cuestión del proyecto fue si mantener los IGBTs o usar transistores MOSFET. Las ventajas de estos últimos son que pueden trabajar a una frecuencia mayor y tienen menos pérdidas de potencia. Lo que se traduce en menor calentamiento. Por otro lado, los IGBTs son más baratos, tenemos disponibilidad de ellos y pueden manejar mayores potencias. Por lo que nos resultó conveniente utilizarlos en nuestro proyecto. En la tabla 3 podemos observar los límites de funcionamiento de los IGBTs elegidos.

IGBT FGH40N60SMDF	
Imáx. colector emisor	40 A
Voltaje de corte	600 V
Temperatura máxima	175 ℃
I pulsante máxima (la duración dependerá de la temperatura)	120 A

Tabla 3 Límites de funcionamiento de los IGBTs

Para la conmutación del IGBT se va a preferir que la carga tenga un comportamiento inductivo. De esta forma la conmutación de paso a ON será ZVS. Por otro lado, en el paso a OFF no podemos evitar las pérdidas en conmutación. Podemos observar el paso a OFF del IGBT en la ilustración 19.

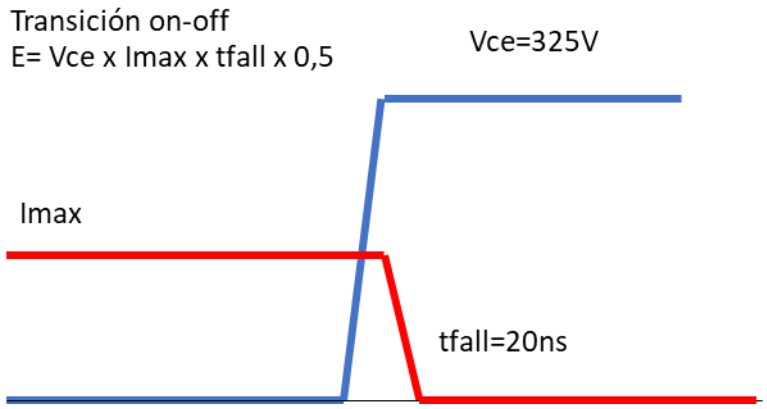


Ilustración 19 Transición de ON a OFF del IGBT

La energía disipada viene determinada por la fórmula que aparece en la ilustración 19. Esta energía será disipada en forma de calor. Se va a proyectar una red de Snubber entre colector y emisor de cada IGBT para ayudar a la conmutación. Debemos dimensionar un condensador que sea capaz de absorber parte de esta potencia, pero sin retrasar excesivamente el tiempo de subida de la tensión. La capacidad óptima para un Snubber de apagado es aquella que absorbe la mitad de la energía [12]. Para su cálculo utilizaremos la ecuación 15.

$$\frac{1}{2} \cdot Csnubber \cdot Vce^2 = Vce \cdot Imax \cdot tfall \cdot 0{,}25 \qquad (15)$$

El término de la izquierda es la energía que almacena el condensador, mientras que el de la derecha es la mitad de las pérdidas de la conmutación. De esta fórmula deducimos que necesitaremos un condensador de 1,23nF. Para sobredimensionar el condensador se ha considerado el valor de la corriente de conmutación máximo de 40A. Obviamente para este

valor de corriente sería necesario reducir la potencia entregada hasta que la corriente eficaz sea inferior a 10A.

El Snubber se instalará solo si comprobamos en la experimentación final que es necesario. A corrientes menores la conmutación se vería ralentizada. Consideraremos a priori que a la frecuencia a la que trabajaremos (cercana a la consumo mínimo) el comportamiento de la carga será prácticamente resistivo.

3.1.3 Disipador térmico

El disipador que se va a implementar posee una resistencia térmica de 0,92℃/W. Lleva incorporado un ventilador interno de 12V. En del disipador colocaremos un sensor de temperatura para poder tener un control de la misma. Las propiedades térmicas del transistor las hemos obtenido de la hoja de datos [13] y aparecen representadas en la tabla 4.

Tmáx. de la unión	175℃
Ice máx.	40A
Runión+Rmica (térmica)	0,63℃/W
Rcond. (Vce/Ice) Vce=2,5V; Ice=40A	0,0625Ω
Fmáx. del inversor (Fijada por nosotros)	200kHz

Tabla 4 propiedades térmicas del transistor

Aunque el transistor sea capaz de aguantar hasta 40A. Nos fijaremos un valor máximo de corriente a 20A. En la ilustración 20 podemos observar el circuito térmico equivalente del disipador. Donde se está considerando que los 4 IGBTs están a la misma temperatura y disipan la misma potencia.

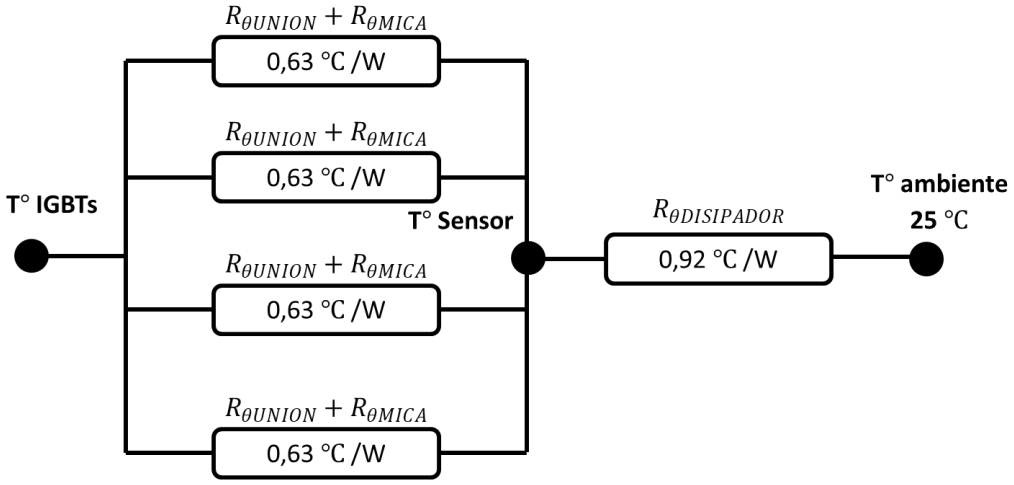


Ilustración 20 Esquema de resistencias térmicas. *Habrá una rama de resistencias en paralelo por cada IGBT. Se considera que los 4 transistores están a la misma temperatura. El sensor de temperatura se puede considerar que esta entre los IGBT y el disipador.*

La potencia disipada (sin Snubber) por cada transistor durante la conmutación a OFF es de 13W considerando un comportamiento fuertemente inductivo. Mientras que en conducción se disiparán 4,13W. El calor total generado por los 4 transistores será de 68,53W. Con estos datos de entrada la temperatura de los IGBTs será de 98,84℃. Entra dentro de nuestros márgenes de operación. Para proteger los IGBTs limitaremos su temperatura de operación a 110℃. El

transistor sigue siendo bastante estable en torno a esta temperatura. Con este parámetro fijado la temperatura que veremos en la posición del sensor será de 97,6℃. Podemos ver el desarrollo para la obtención de las temperaturas en el anexo de cálculos.

3.1.4 Comunicación serie por fibra óptica

Las señales de activación de los IGBT y la realimentación para el control viajan por el exterior de la placa hasta un microprocesador (Arduino Due). El cuál manda sobre el funcionamiento del inversor. Al viajar por el exterior de la placa estas señales son vulnerables al ruido electromagnético que produce la bobina de Tesla. Si la señal se ve alterada podrá generar desde problemas en el control hasta la incorrecta activación de los IGBTs.

Para evitar las interferencias externas la comunicación entre placa y procesador se hará por fibra óptica. La mayor ventaja es que tanto el sistema de control, como la persona que en su caso lo deba modificar, se hayan a una distancia segura de la bobina de Tesla. Los componentes escogidos son el HFBR-2521 como receptor y el HFBR-1521 como transmisor. La señal de activación de los IGBT tendrá una conexión de recepción por rama del inversor. Para la realimentación habrá una conexión de recepción y otra de transmisión (aunque nuestro diseño solo será de transmisión). Estas conexiones llegarán a un Arduino Nano que actuará como módulo de comunicaciones, a través de su UART interna. El diagrama será como el esquema de pulsos generados externamente de la ilustración 17.

3.1.5 Protección del IGBT

Los componentes más importantes del inversor son los transistores IGBT. Si alguno falla el puente dejará de funcionar. Como medida de protección adicional se ha protegido galvánicamente el circuito de disparo. Un error de disparo puede producir un cortocircuito dañando irreversiblemente la placa. Podemos resumir las causas de destrucción del transistor en dos:

- Pico de corriente: Si un pulso supera los 120A el transistor quedará deteriorado.
- Exceso de temperatura: Si la temperatura supera los 125℃ el transistor se degradará y tal como indica el fabricante por encima de 175℃ quedará inoperativo.

Estos son valores de funcionamiento límite. Los valores de operación recomendados por el fabricante son 100℃ y 40A de colector a emisor. Daremos un margen a la temperatura de hasta 110℃. La respuesta de temperatura es mucho más lenta en comparación a la de corriente. Su limitación será controlada directamente por el Arduino Nano. El cuál inhabilitará los pulsos de control si esta supera el valor máximo definido. Hasta que la temperatura caiga por debajo de un valor de histéresis. El sensor elegido es el LM35 con una sensibilidad de 10mV/℃. Estará ubicado dentro del disipador. En el apartado 3.1.3 vimos que la temperatura máxima en el sensor deberá ser de 97,6 ℃. La temperatura de histéresis la fijaremos a 75 ℃.

Respecto a la corriente, el valor límite que asegura un correcto funcionamiento del transistor es de 40A. A partir de este valor puede existir un pulso lo suficientemente largo que dañe irreversiblemente el IGBT. El objetivo es proteger contra pulsos de corriente que puedan darse de manera puntual sobrepasando el límite constructivo. La protección debe reaccionar lo suficientemente rápido cuando una sobrecorriente sea detectada. Funcionará de manera asíncrona, no es controlada por el Arduino. Cuando una corriente active la protección. Esta desconectará los pulsos de disparo y avisará al Nano. El diseño de la protección consiste en un amplificador operacional sin realimentación, el cuál comparará entre una consigna de tensión preestablecida, en función de la corriente máxima, y la tensión de una resistencia de shunt de

0,1Ω. Colocada en la rama inferior del puente. Habrá un módulo de protección por rama del inversor. Destacamos que la consigna de tensión tendrá un control manual. Como máximo la fijaremos para una corriente de 40A, pero podemos limitarla a valores más pequeños según nos convenga. En la ilustración 21 se puede observar el esquema eléctrico del limitador.

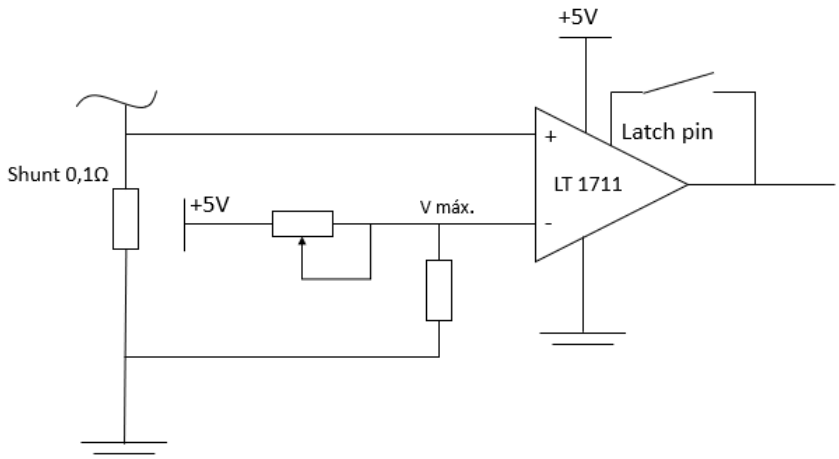


Ilustración 21 Esquema de funcionamiento del limitador de corriente

El amplificador elegido es el LT1711. Es un comparador de alta velocidad rail to raíl [14]. Carece de aislamiento galvánico. Este comparador posee un pin de latch. Si el valor de este es 1 se guarda el valor de la salida. La conexión está hecha para que cuando el comparador pase a un nivel alto el latch también lo haga. Esto nos asegura que el inversor se detendrá hasta que el Arduino rearme la protección actuando sobre el interruptor de la ilustración 21, el cual eliminará el valor del latch cuando detecte que no hay corriente por el inversor. Destacamos que la configuración del interruptor es normalmente cerrada.

3.2 Construcción del prototipo

Una vez definidos los componentes más importantes que utilizaremos y con los planos ya diseñados pasaremos al montaje de la placa. Los planos se podrán revisar en el anexo de planos. Los componentes pasivos utilizados han sido escogidos según las recomendaciones de los fabricantes o por los valores típicos para la función que desempeñan. En cuanto a los componentes activos, como los reguladores de tensión, estos no tienen ninguna prestación especial añadida. Son componentes comunes y con disponibilidad en el departamento. Se ha tratado de hacer el diseño más didáctico posible. Añadiendo varios testpoints y testigos luminosos.

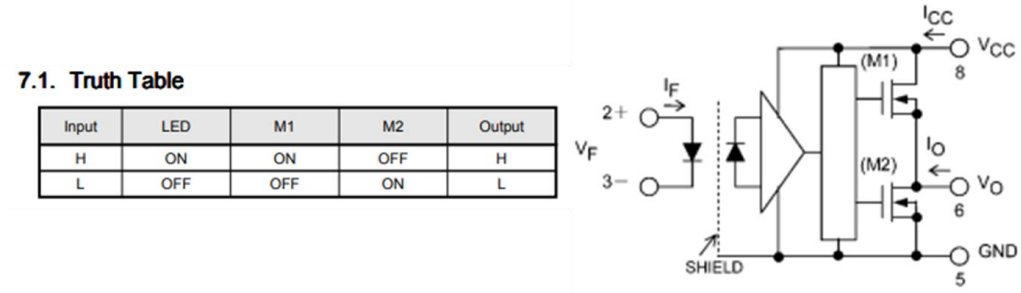
3.2.1 Alimentación a 5 y 12V

Los componentes activos necesitan alimentarse a tensiones de 5 V y 12 V de continua. Será necesario darles un valor de tensión estable. El circuito que se diseñará es de baja potencia. La alimentación irá completamente separada de la alimentación del puente inversor.

Se usará un transformador reductor 1:12 de dos devanados. Los puentes de diodos son el B250C1500. El regulador de tensión para generar los 12V es el L7812. Mientras que para generar 5V el componente es el L7805. Con la finalidad de hacer un circuito más intuitivo en los ensayos, se ha añadido un led verde donde haya una alimentación de 5V y uno rojo para 12V.

3.2.2 Módulo de disparo del IGBT

Se ha diseñado un módulo intercambiable para adaptar los disparos del IGBT. Está formado por un regulador de tensión elevador de 12V a 15V NME1215SC. Posee aislamiento galvánico entre su entrada y salida. Alimenta los componentes activos de este módulo. Mientras que el componente TLP250HF se encarga de activar el IGBT según los pulsos que le lleguen. Su principio de operación lo define la tabla de la verdad que aparece en la ilustración 22.



7.1. Truth Table

Input	LED	M1	M2	Output
H	ON	ON	OFF	H
L	OFF	OFF	ON	L

Ilustración 22 Principio de operación del TLP250HF. *[15]*

Este componente posee aislamiento galvánico entre su puerta y la señal de activación. La salida de este driver pondrá en +15V y 0V la tensión base-emisor del IGBT. Tensión suficiente para que el transistor de potencia entre en conducción. La referencia negativa de estos módulos es flotante. Está ubicada en el emisor de su IGBT correspondiente. En la ilustración 23 se observa el esquema eléctrico de este módulo.

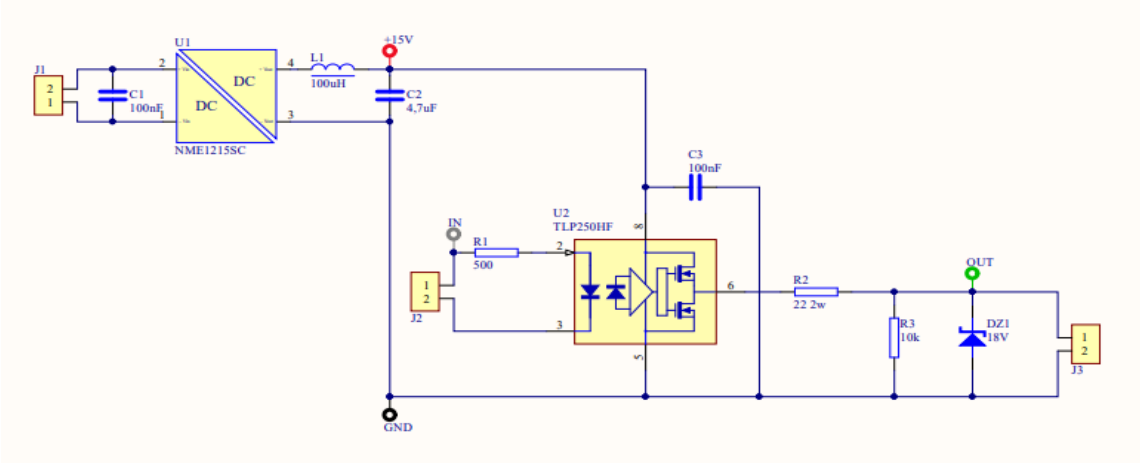


Ilustración 23 Esquema de disparos del IGBT. *La alimentación de 12V llega al J1 en la parte superior izquierda. Los pulsos de la fibra óptica convertidos ya en pulsos de tensión llegan al TLP250HF a través de J2. Este transistor activa un circuito de disparo que llega a J3 en forma de 15V o 0V.*

3.2.3 Sensores y elementos de protección activa y pasiva

Las protecciones activas protegerán frente a excesos de corriente y temperatura. En el caso de la sobretemperatura la protección dependerá del sensor. Las protecciones pasivas instaladas son los fusibles de entrada que soportan hasta corrientes de 10A eficaces. También se colocarán descargadores en la entrada y salida de potencia del sistema para prevenir contra sobretensiones puntuales. Se instalarán 3 sensores encargados de tomar los valores de tensión del bus de continua, de corriente justo antes del condensador del circuito inversor y la temperatura de los IGBTs. En este apartado nos centraremos los sensores de voltaje, corriente y protección contra sobrecorrientes.

Voltaje del bus de continua

El sensor analógico utilizado es un divisor de tensión. Una resistencia de 68kΩ y otra de 820Ω capaces de soportar hasta 3W de potencia. El divisor atenúa la tensión del bus en un factor de 0,012 Vsensor/Vbus. El sensor tiene de rango de medida entre 0V y 420V. Tomando valores entre 0V y 5V. La salida del sensor irá conectada a un pin analógico del Arduino. De esta medida se puede obtener el voltaje DC e incluso el rizado. En la ilustración 24 podemos ver el equema de aplicación de este sensor.

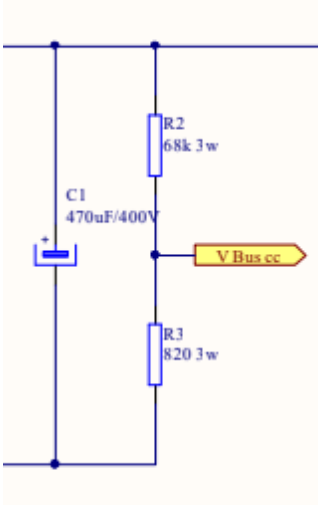


Ilustración 24 Esquema del sensor de voltaje

Sensores de corriente

Se utilizará un sensor Hall HLSR-20P. Según su hoja de datos son capaces de medir desde 50A hasta -50A en un ancho de banda de 400kHz [16]. Se colocará en la entrada del condensador del puente inversor. Por este sensor circulará corriente solo cuando los componentes reactivos, como el condensador rectificador, necesiten recargarse debido a las pérdidas de potencia activa. La corriente que midamos aquí nos permitirá calcular la potencia activa que se entrega al sistema. Podemos ver el esquema del sensor Hall en la ilustración 25.

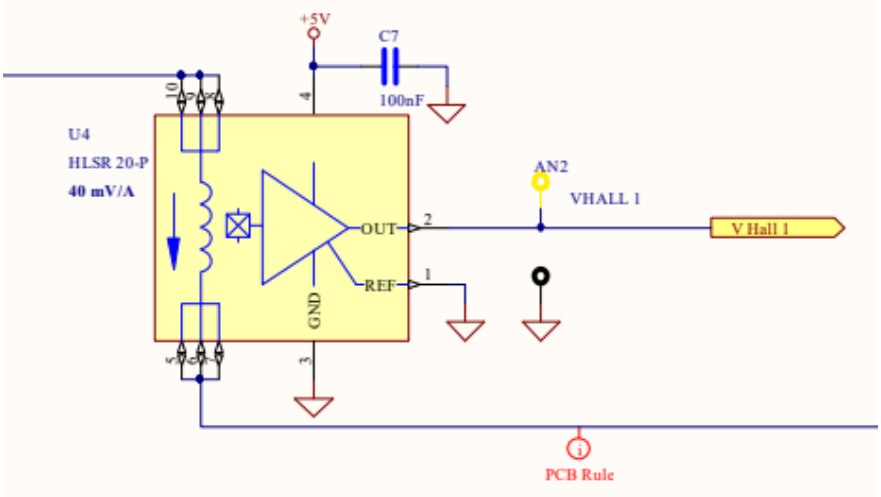


Ilustración 25 Esquema del sensor Hall

Protecciones

Hablaremos solo de la implementación del módulo de protección contra sobreintensidades, debido a una peculiaridad encontrada en su funcionamiento. Podemos observar el esquema de aplicación en la ilustración 26.

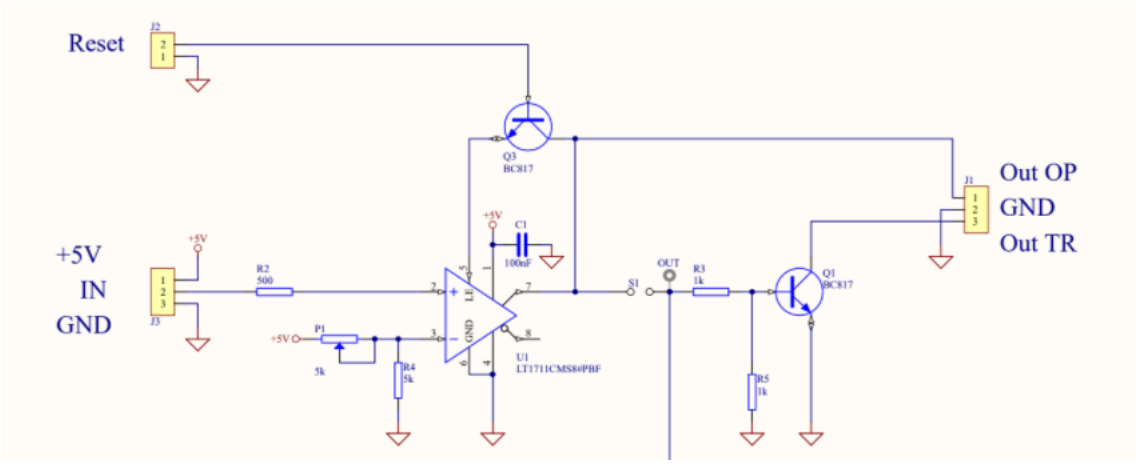


Ilustración 26 Esquema del comparador

Los dos módulos están ubicados entre el emisor del IGBT y el neutro del inversor, como ya se pudo observar en la ilustración 21. Durante la prueba del módulo sobre la placa principal se pudieron observar fuertes rebotes durante las inversiones de tensión del puente. Los rebotes eran tales que activaban la protección sin que el nivel de corriente fuese sobrepasado. Se pudo determinar que se debía a la inductancia parásita de la resistencia de shunt. Según su datasheet está en torno de los 10nH. Lo que ocurre es que el tiempo de paso a OFF del IGBT es tan rápido que la derivada de la corriente que circula en el paso a OFF es muy elevada. Por lo que producirá fuertes oscilaciones entre los extremos del shunt medibles por el comparador. La solución que se adoptará será la calibración del nivel de disparo de la protección a un valor por encima de los rebotes. Con la finalidad de solucionar el problema con las herramientas disponibles. La protección de sobrecorrientes es uno de los elementos a mejorar en líneas futuras.

Arduino Nano

El Arduino Nano actuará como módulo de comunicaciones. Recibirá la información de todos los sensores de la placa. Realizará los cálculos y enviará una realimentación al procesador principal. A parte de la lectura de las señales se le ha añadido conexiones que permiten que este sea más interactivo para el usuario como pulsadores o una pantalla de lectura. A continuación, veremos el mapa de conexiones en la tabla 5.

PIN	ETIQUETA	DESCRIPCIÓN
RX0	RX	Recepción de datos con el procesador externo a través de fibra óptica.
TX1	TX	Transmisión de datos con el procesador externo a través de fibra óptica.
D2	Fallo comparador shunt 1	Entrada conectada a la salida de uno de los comparadores, activará una interrupción.

D3	Fallo comparador shunt 2	Entrada conectada a la salida de uno de los comparadores, activará una interrupción.
D8	Pulsador 1	Entrada digital conectada a un pulsador externo
D13	Pulsador 2	Entrada digital conectada a un pulsador externo
RST	Pulsador 3	Entrada digital conectada a un pulsador externo
D4	Pulsador 4	Entrada digital conectada a un pulsador externo
D9	Reset shunt 1	Salida digital conectada al reset de uno de los comparadores. Resetea este en nivel bajo
D5	Reset shunt 2	Salida digital conectada al reset de uno de los comparadores. Resetea este en nivel bajo
D6	INH pulsos	Salida digital conectada a la línea de pulsos del IGBT. En nivel alto los inhibe.
D7	Relé precarga	Salida digital conectada al relé de precarga. En nivel alto abre el relé.
D10	Pulsos ventilador	Salida conectada al control del ventilador.
A0	Tensión bus de continua	Entrada analógica para la lectura de la tensión del bus de continua.
A1	Tensión sensor de temperatura	Entrada analógica para la lectura de la temperatura de los IGBT.
A2	Tensión sensor Hall 1	Entrada analógica para la lectura de la corriente.
A3	Tensión sensor Hall 2	Entrada analógica para la lectura de la corriente.
D11	Pulsos A	Entrada digital para la lectura de los pulsos de disparo.
D12	Pulsos B	Entrada digital para la lectura de los pulsos de disparo.
A4	SDA	Comunicación I2C con pantalla.
A5	SCL	Comunicación I2C con pantalla.

Tabla 5 Mapa de conexiones del Arduino Nano

Pulsos de control de los IGBTs

En la placa principal de potencia se ha reservado un espacio donde se conectará un módulo que genera los disparos del IGBT. En este espacio se pueden colocar dos módulos. El módulo generador de pulsos analógico. El cuál tiene un control manual para regular frecuencia, dead time y duty cycle. Para ello utiliza el controlador SG3525AN [17] heredado del diseño anterior. La otra opción es conectar el módulo de fibra óptica. Este actúa como receptor de

disparos generados externamente. Estas dos configuraciones nos permiten generar los pulsos interna o externamente. Se eligió así para poder tener un control manual durante las pruebas de funcionamiento. Además, el módulo generador pulsos analógico ya ha sido probado en el prototipo anterior, por lo que en este apartado solo hablaremos del módulo de señales por fibra óptica. Podemos observar su esquema eléctrico en la ilustración 27.

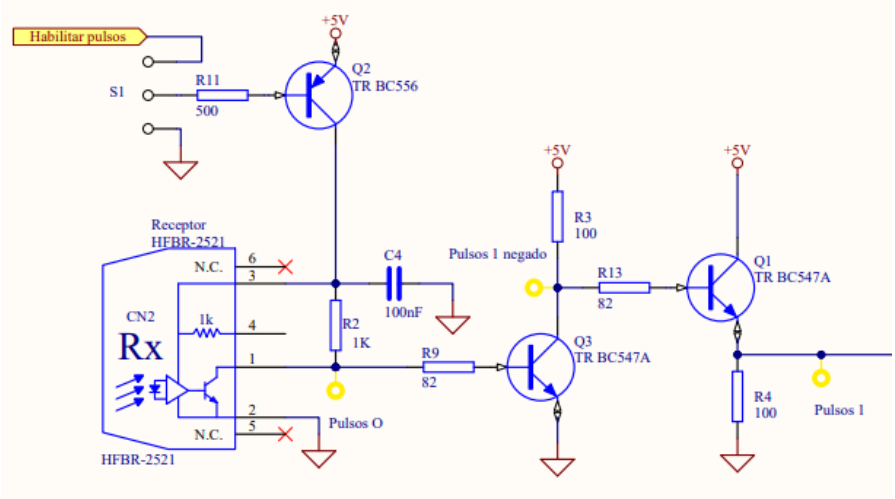


Ilustración 27 Esquema de disparos por fibra óptica

En esta parte se convierte el haz de luz que circula por la fibra en una señal de tensión cuadrada que pueda disparar los IGBTs. El transistor superior es el encargado de inhibir los pulsos cuando se le dé la orden. La idea es que este módulo sea intercambiable entre la generación interna y externa de los pulsos utilizando para ellos las mismas entradas y salidas.

PLACA DE POTENCIA

La placa de potencia es aquella en la que se ubica el puente inversor. Es la parte más grande. En ella se encuentran todos los módulos intercambiables. Sobre ella también está ubicado el circuito que genera la alimentación de 5V y 12V del que ya hemos hablado anteriormente. Ambos circuitos están completamente aislados. La placa también soporta los sensores nombrados anteriormente. Los componentes principales de la placa de potencia ya han sido enumerados anteriormente (rectificador de diodos trifásica, IGBTs, etc.) La operatividad del puente inversor depende de los módulos conectados. Estos vienen enumerados en la tabla 6.

MÓDULOS	
MÓDULO DE DISPAROS IGBT	Adapta la señal analógica de disparo a la señal que necesita el IGBT. Posee aislamiento galvánico.
MÓDULO COMPARADOR	Detecta los picos de corrientes en las ramas del inversor.
MÖDULO DE ARDUINO NANO	Actúa como módulo de comunicaciones con el exterior.
MÓDULO GENERADOR DE PULSOS ANALOGICO	Genera dos señales cuadradas regulables manualmente. Puede ir ubicado interna o externamente.
MÓDULO DE IMPULSOS POR FIBRA	Actúa como receptor de un módulo generador de señales externo. Es intercambiable con el módulo generador de impulsos.

Tabla 6 Módulos de la placa principal

4. Diseño del Firmware

En este apartado hablaremos de los códigos implementados para el control del puente inversor. Podemos diferenciar entre la programación del Arduino Nano y Due. El Arduino nano estará ubicado sobre la placa principal del puente inversor. Hará las labores de módulo de comunicaciones para un procesador externo. Este último se encargará de interpretar la realimentación del procesador interno y generar una respuesta en función de esta lectura para modificar la salida del inversor. El Arduino Due también hará las labores de interfaz hombre máquina. En el anexo firmware se encuentran todos los programas utilizados.

4.1 Arduino Nano

Este microprocesador tiene la tarea de sensar las señales deseadas. Procesar estos valores para obtener los parámetros necesarios. Para después enviar estos datos a través de un cable de fibra óptica al procesador externo. Destacamos que el Arduino Nano estará enviando constantemente los datos de funcionamiento de la placa. Con un período aproximado en torno a los 100ms independientemente de si alguien los está leyendo o no. La comunicación es solo de transmisión. El bucle de control de este módulo sigue el orden de la ilustración 28.

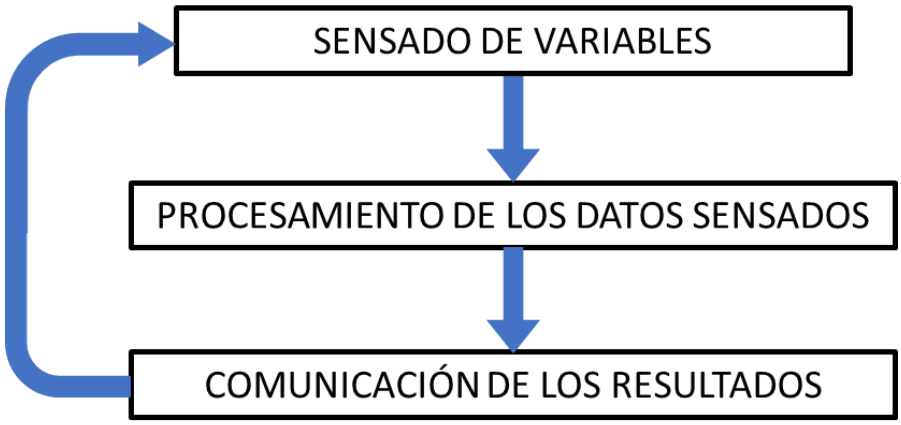


Ilustración 28 Bucle de control del microprocesador interno. *El bucle se reinicia 50ms después de la comunicación de los resultados. El sensado de variables tarda aproximadamente 30ms. El tiempo de ejecución de los bloques restantes en torno a 20ms más.*

SENSADO DE VARIABLES

Este procedimiento dentro del bucle principal se encarga de sensar los valores de tensión del bus, la corriente de carga del condensador rectificador y la temperatura del IGBT. El sensado se realiza a una velocidad de 10kS/s a través de tres puertos analógicos conectados al ADC de 10 bits del Arduino Nano.

El proceso comienza tomando 150 lecturas de la corriente del sensor Hall ubicado antes del condensador. Circulará corriente solo cuando el valor de la tensión del bus esté por debajo de la tensión de red. El siguiente paso es tomar 150 lecturas de tensión del bus de continua en el divisor de tensión, nombrado en el capítulo anterior, que utilizamos como sensor de voltaje. Por último, se tomará solo una lectura de la temperatura por ciclo del bucle. Se considera que la temperatura variará muy lentamente.

PROCESAMIENTO DE SEÑALES Y ALERTAS

En este apartado veremos cómo se procesan los valores sensados anteriormente para calcular la tensión del bus, la corriente de carga, la potencia y la temperatura. También veremos cómo procesa el Arduino las alertas de sobrecorrientes.

Empezaremos hablando del cálculo de la corriente de carga. Se considerará la aproximación que aparece en la ilustración 29.

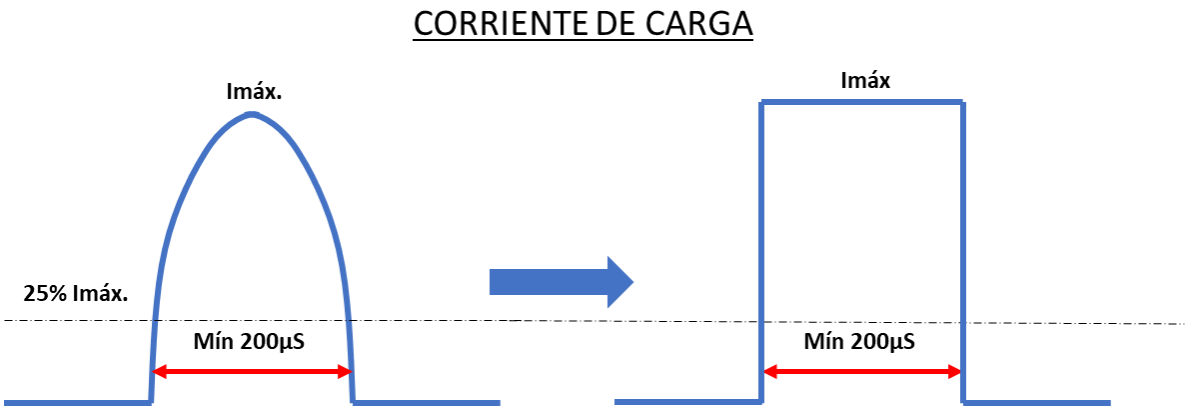


Ilustración 29 Aproximación de la corriente de carga. *La corriente de carga tiene la forma de la izquierda. Se va a aproximar por un pulso cuadrado como el de la derecha.*

El pulso de carga debe ser como mínimo de 200µs (el tiempo de dos conversiones). Para que el ADC sea capaz de leerlo. La anchura del pulso depende del rizado de la tensión del bus. Para el valor mínimo de anchura de pulso el rizado será del 0,2%. Para extraer la anchura de pulso y la corriente del pulso el proceso será el siguiente:

1. Se buscará el valor máximo que toma la corriente
2. Se buscará el primer instante en el que la corriente supere el 25% del valor máximo. Este será el flanco de subida del pulso.
3. Inmediatamente se buscará el instante en el que la corriente baje del 25% del valor máximo. Definiendo así el flanco de bajada y por tanto la anchura del pulso.
4. La corriente del pulso se aproximará por el valor máximo. Siendo esta una aproximación mayorada pero válida porque estamos considerando los flancos de subida al 25% de la señal. Disminuyendo la magnitud del tiempo de carga.

Una vez procesados los valores del sensor Hall pasaremos a procesar el voltaje. Para ello se ha considerado que la tensión tiene la forma de la ilustración 30.

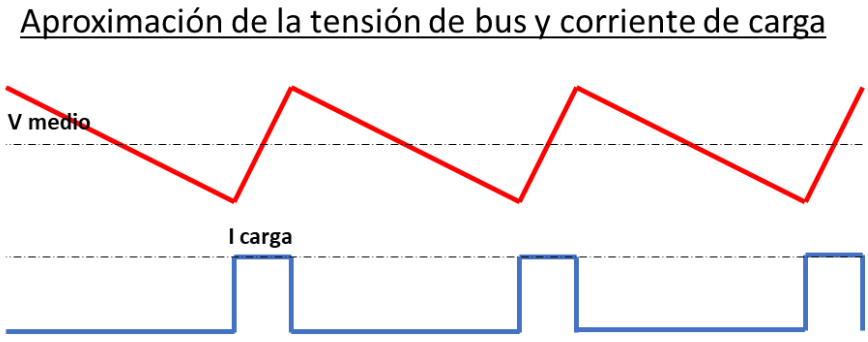


Ilustración 30 Diagrama del voltaje de bus y corriente de carga

Para extraer el valor medio de la tensión se realizará el siguiente proceso:

1. Buscamos el valor máximo de la tensión del bus.
2. Buscamos el valor mínimo de la tensión del bus.
3. Realizamos la media entre las dos.

Respecto a la conversión de la temperatura la medida es directa. Solo tendremos que convertir la lectura del ADC a las unidades correspondientes. Si la temperatura supera el valor límite generará una alerta que será procesada más adelante. Para la conversión de los parámetros a las unidades correspondientes utilizaremos las ecuaciones 15, 16 y 17.

$$I_{CARGA}[A] = (\frac{5}{1024} \cdot I_{ADC} - 2{,}5) \cdot (\frac{50}{2}) \quad (16)$$

$$V_{PROMEDIO}[V] = \frac{5}{1024} \cdot V_{ADC} \cdot \frac{68820}{820} \quad (17)$$

$$T_{IGBT}[°C] = \frac{5}{1024} \cdot T_{ADC} \cdot 0{,}01 \quad (18)$$

Por último, para el cálculo de la potencia utilizaremos los valores de corriente y voltaje calculados anteriormente. La potencia se modela con la ecuación 19.

$$P[W] = V_{PROMEDIO} \cdot I_{CARGA} \cdot (k_{subida} - k_{bajada}) \cdot \frac{2 \cdot fred}{fsensado} \quad (19)$$

Donde k_{subida} y k_{bajada} son los instantes discretos de los flancos de subida y bajada. La frecuencia de sensado es 10kHz. La frecuencia de red se multiplicará por 2 en la configuración monofásica y por 6 en la trifásica. Respecto a las sobreintensidades debemos saber que las salidas de los comparadores están conectados a los pines de interrupciones externas del Arduino Nano. Cuando una sobrecorriente es detectada se activará una alerta que es procesada más adelante.

COMUNICACIONES

La última parte del bucle de control es el apartado de comunicaciones aquí se transmiten los resultados de las medidas por el puerto serie al Arduino Due y por comunicación I2C a una pantalla LCD. En la ilustración 31 podemos ver el diagrama de flujos que utiliza la comunicación.

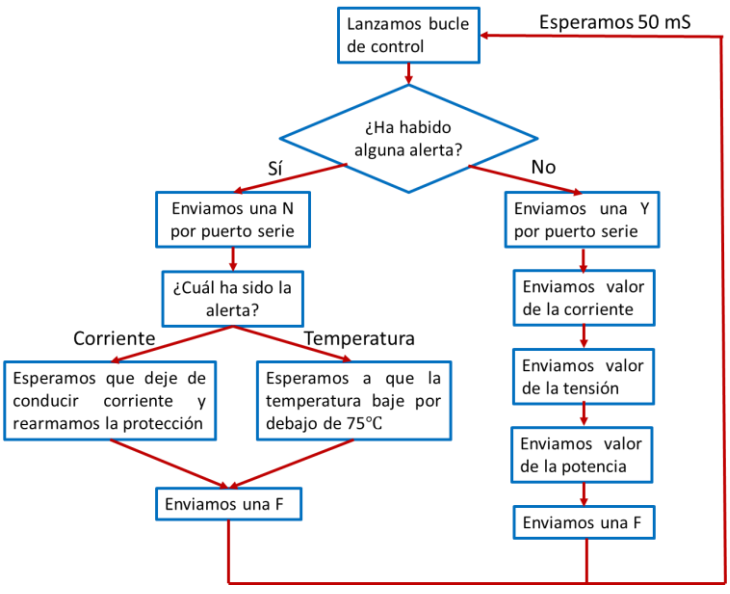


Ilustración 31 Diagrama de flujos de la comunicación

Por puerto serie veremos una columna de valores como la de la tabla 7. Lo que veamos dependerá del funcionamiento. Se enviará un mensaje cada 100ms aproximadamente (50ms para el procesamiento del programa y 50 ms de espera entre bucles de control)

FUNCIONAMIENTO CORRECTO	FUNCIONAMIENTO INCORRECTO
Y	N
Valor de I carga	1/2/3 (1 y 2 significan alertas de corriente, 3 de temperatura)
Valor de V de bus	F
Valor de Potencia	-
F	-

Tabla 7 Mensajes por puerto serie

Los mensajes empezarán con una "N" o "Y" dependiendo si se ha activado una protección o no. Todos los mensajes terminan con una F. La comunicación con la pantalla LCD sigue un comportamiento similar. En la pantalla podemos ver los valores de corriente de carga, potencia, temperatura y voltaje de bus. Si se ha activado alguna protección se mostrará por pantalla el número de la alerta.

4.2 Arduino Due

El microprocesador conectado a un ordenador a través del puerto serie. Sirve de interfaz para el usuario final y también de control del puente inversor. Por el puerto serie del ordenador se mostrará un menú principal que dejará elegir entre 3 submenús para determinar el control deseado. Las funciones que se pueden elegir son las siguientes:

- Submenú para la modificación de los valores predeterminados del dead time entre canales y del duty cycle.
- Submenú para fijar una frecuencia de funcionamiento. Si se elige esta función veremos por puerto serie la respuesta del Arduino Nano. En caso de activarse una protección se apagarán los timers y regresaremos al menú principal.
- Submenú para buscar la frecuencia de consumo mínimo de potencia.

Los menús han sido diseñados para que sean lo más interactivos posibles para el usuario final con la finalidad de facilitar el entendimiento del programa.

Las funciones más importantes implementadas en el Arduino Due son las encargadas de fijar una frecuencia y la encargada de buscar el valle del consumo de la potencia.

Función para fijar la frecuencia del inversor

Esta función denominada "set_frequency" tiene como valores de entrada la frecuencia de excitación del puente, el ciclo de trabajo del 1 al 100%, el dead time entre canales y el control por ráfagas del 1 al 100%. La función para la modulación del ciclo de trabajo fue introducida para dar más funcionalidades a la placa. Para la aplicación de la bobina de Tesla el ciclo de trabajo es del 100% (50% por rama del inversor).

El dead time fija un valor de tiempo mínimo entre la alternancia de pulsos de ambos canales. Evitando activaciones simultáneas de los IGBTs de una rama. El dead time es un tiempo que se resta al ON del transistor alargando su tiempo en OFF. Por lo que deberá ser lo más pequeño posible. El dead time mínimo que podemos fijar con el Arduino Due es de 300ns. El valor que fijemos tendrá una tolerancia de ±10ns. El control del duty cycle se hará aumentando directamente el valor del dead time para que los transistores permanezcan apagados la fracción de tiempo indicada. Destacamos del ciclo de trabajo que por canal como máximo del 50%. (restando el valor del dead time mínimo) Pero al tener 2 canales que se alternan entre sí, el ciclo de trabajo total será el doble.

El control por ráfagas dejará pasar tantos pulsos de cada 100 como hayan sido indicados en la entrada. Las salidas de la función "set_frequency" serán las salidas de 2 timers del Arduino Due con los parámetros fijados previamente. Podemos observar cómo es la respuesta en la ilustración 32.

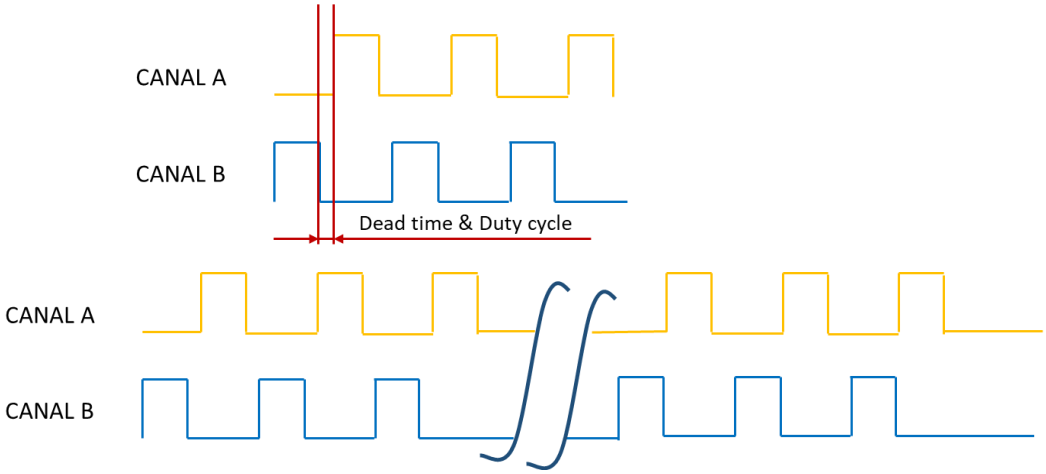


Ilustración 32 Respuesta de la función "set_frequency". *La imagen superior muestra el control del duty cycle junto a la del dead time. Mientras que la imagen inferior muestra el control por ráfagas*

Función para buscar el valle de la potencia

Esta función denominada "buscar_valle" tiene como entrada la frecuencia inicial desde la cual se empezará a buscar el valle de la potencia. Apoyándose en la función "set_frequency" se realizarán recorridos en torno a la frecuencia inicial buscando el punto con menor potencia del sistema. Su principio de funcionamiento aparece en la ilustración 33.

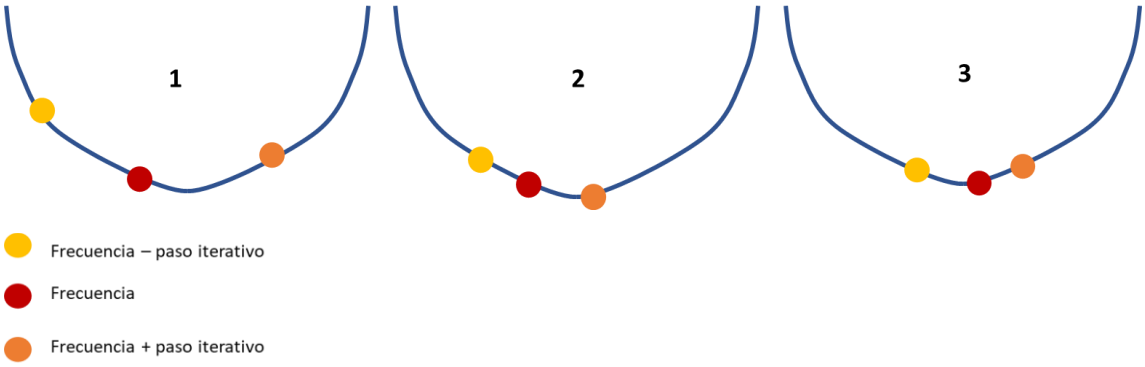


Ilustración 33 Respuesta de la función buscar valle. *Las curvas que observamos son las respuestas de la potencia en función de la frecuencia. El punto rojo es la frecuencia de partida de cada iteración. Se puede observar que del paso 1 al 3 nos iremos aproximando cada vez más al valle.*

La frecuencia de partida de cada iteración es el punto rojo de la ilustración 33. Para elegir la frecuencia inicial de la próxima iteración nos basamos en el punto de menor potencia. Si este punto es el mismo que el anterior. El paso de la frecuencia disminuirá. Conforme nos acercamos al valle el paso iterativo ira disminuyéndose. Empezará siendo 1600Hz, pasará a 800Hz, después a 400Hz y 200Hz para acabar siendo 100Hz. Cuando el paso llega a 50Hz se considera que hemos encontrado el valle de potencia. Cada iteración recorre 3 frecuencias. Una por encima y otra por debajo de la frecuencia de partida distanciadas un paso respecto del valor inicial. Como máximo podrá haber hasta 300 iteraciones. Si se diese el caso de que las tres frecuencias de la iteración activan a la vez una protección de sobreintensidades o al menos una active la protección de temperatura el programa se cancelaría y regresaría al menú principal.

5. Ensayos finales

En este apartado se muestra el resultado de los ensayos realizados con el sistema implementado sobre una bobina de Tesla real. Uno de los objetivos es caracterizar alguno de los parámetros más importantes de ésta. Por ejemplo, las frecuencias de resonancia y la potencia. Antes de ello se realizarán las pruebas pertinentes sobre los componentes de la placa para comprobar que responden correctamente.

5.1 Verificación del funcionamiento

Lo primero que se verificó fue si la señal de control llegaba correctamente a la base de los IGBTs. Dada una frecuencia establecida con el controlador manual. Se comprobó que la señal de control era la esperada. Podemos ver el montaje utilizado para las pruebas de potencia en la ilustración 34.

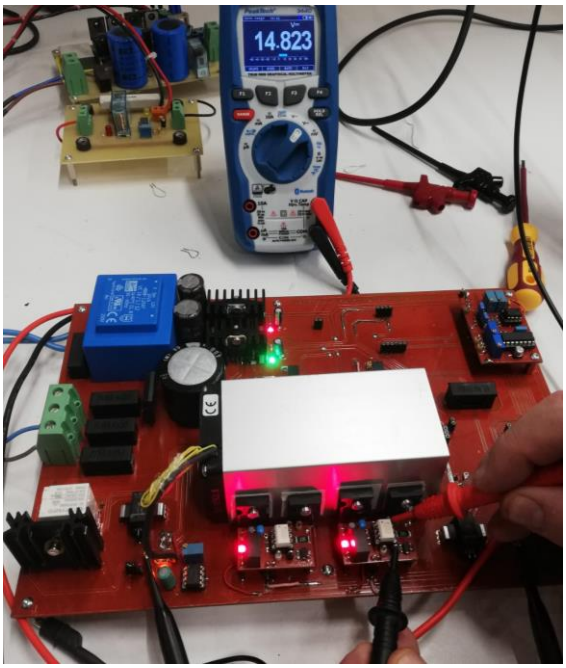


Ilustración 34 Pruebas sobre la placa de potencia.

Una vez comprobado el funcionamiento de los drivers del IGBT se procedió a realizar las primeras pruebas de potencia. Se conectó una carga de 25Ω y 75µH a 77kHz y 50V. En estas pruebas el control era en bucle abierto. Dábamos la consigna de frecuencia manualmente a través del módulo generador de pulsos analógico. Sin utilizar la conexión de fibra óptica. En esta

parte tampoco utilizamos las protecciones, solo se pretendía verificar que el inversor respondía correctamente. Los resultados de esta prueba se pueden observar en la ilustración 35.

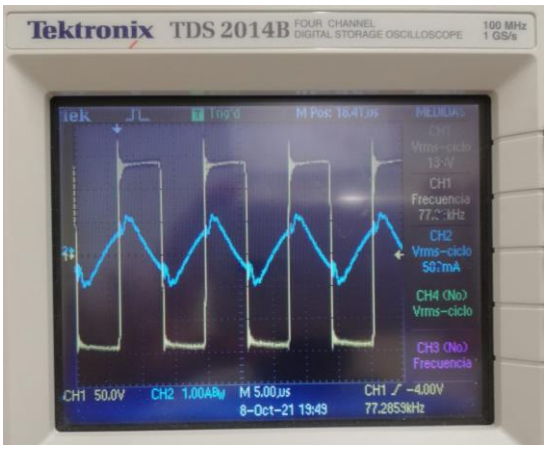


Ilustración 35 Señales de salida. *Azul corriente, amarillo tensión. La corriente esta invertida*

La conclusión es que nuestro prototipo es capaz de rectificar e invertir la tensión de entrada a la frecuencia que se le especifique. La posterior conexión de los módulos no afectará a esta funcionalidad, estos son necesarios para poder ejecutar un control del funcionamiento.

El siguiente paso fue comprobar que la respuesta de la ilustración 35 se repetía al generar los impulsos con un Arduino Due y enviar estos a través de un cable de fibra óptica. Podemos ver la PCB diseñada para el Due en la ilustración 36.

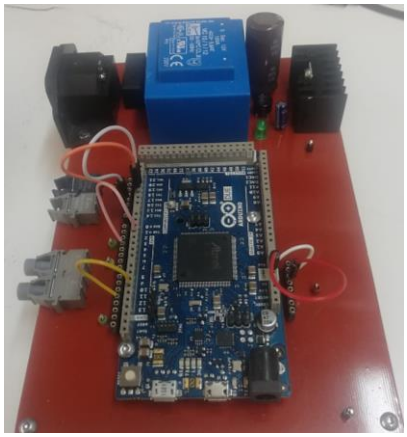


Ilustración 36 PCB para Arduino Due

Después de comprobar que el puente por fibra óptica también funcionaba correctamente. La conclusión fue que nuestro prototipo también es apto para el control externo. Respecto a los elementos para el control como los sensores, Arduino Nano y protecciones contra sobrecorrientes podemos decir lo siguiente.

- El sensor de temperatura y de voltaje responden correctamente.
- El sensor Hall no pudo estar operativo para las pruebas realizadas.
- Las protecciones contra sobrecorrientes detienen el sistema cuando la tensión de su resistencia de shunt supera al valor de referencia calibrado.
- El módulo para conectar el Arduino Nano que se puede observar en la ilustración 37. No pudo estar operativo para las pruebas finales por problemas del conexionado dentro de la placa. Su comunicación con la pantalla LCD y Arduino Due se pudo probar con una señal virtual. El firmware para el sensado, cálculo y

comunicación del Arduino Nano fue probado con señales de comportamiento conocido creadas en un banco de pruebas, verificando de esta forma que los códigos diseñados funcionarán cuando el módulo este plenamente operativo.

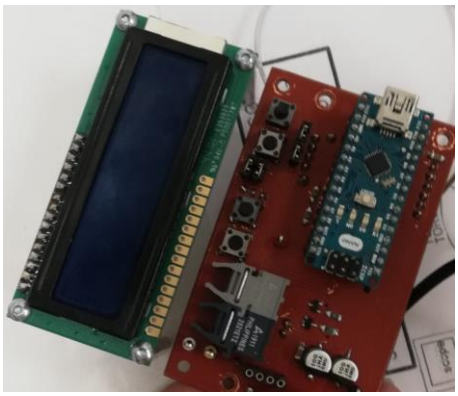


Ilustración 37 Módulo del Arduino Nano. *La pantalla ira conectada encima de él.*

Como el Arduino Nano no estaba listo para las pruebas sobre la bobina de Tesla. Se efectuará un control en bucle abierto. En el que iremos variando la consigna de la frecuencia a través del puerto serie del ordenador para poder caracterizar la bobina. El programa utilizado para el Arduino Due aparece en el anexo firmware.

5.2 Pruebas sobre la bobina de Tesla

Con la placa operativa en lazo abierto, se va a realizar un estudio experimental en régimen permanente sobre una de las bobinas de Tesla del laboratorio. Esta bobina es la que habíamos modelado previamente en Matlab. Las características de nuestra placa aparecen en el anexo hoja de datos. El objetivo será obtener de forma experimental sus tres frecuencias características y la curva de potencia aparente de la red. También queremos observar la alternancia de comportamiento inductivo a capacitivo. En la ilustración 38 podemos observar el montaje utilizado para la realización de las pruebas.

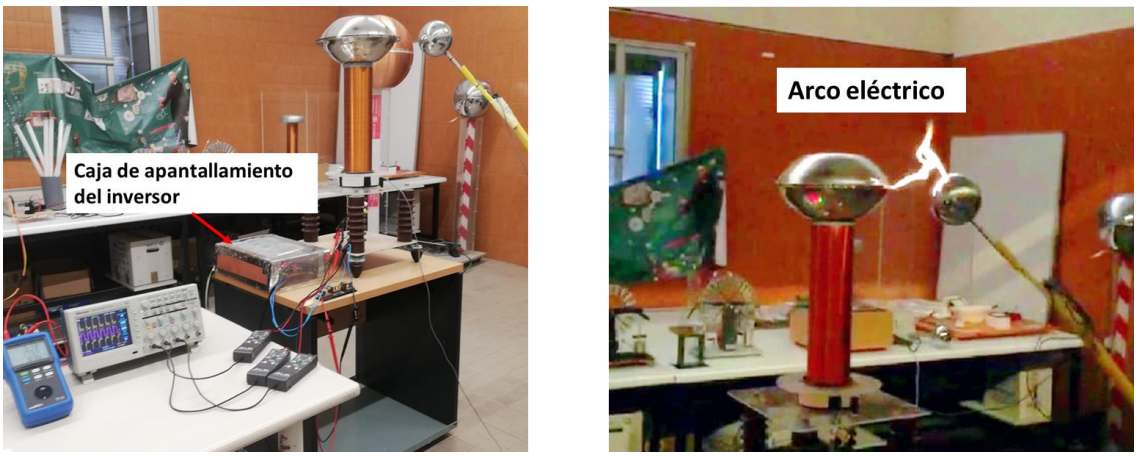


Ilustración 38 Montaje para los ensayos sobre la bobina de Tesla. *En la figura de la izquierda se observa el montaje utilizado. A la derecha se observa un arco eléctrico generado a 149kHz.*

El ensayo se hará inicialmente a una tensión de 15V para poder ampliar nuestro rango de operación, sin control de potencia y con un ciclo de servicio del 50% por rama. El dead time mínimo se ha fijado a 400ns, la alimentación es monofásica y no se han producido descargas al ambiente. La placa final del puente inversor aparece en la ilustración 39.

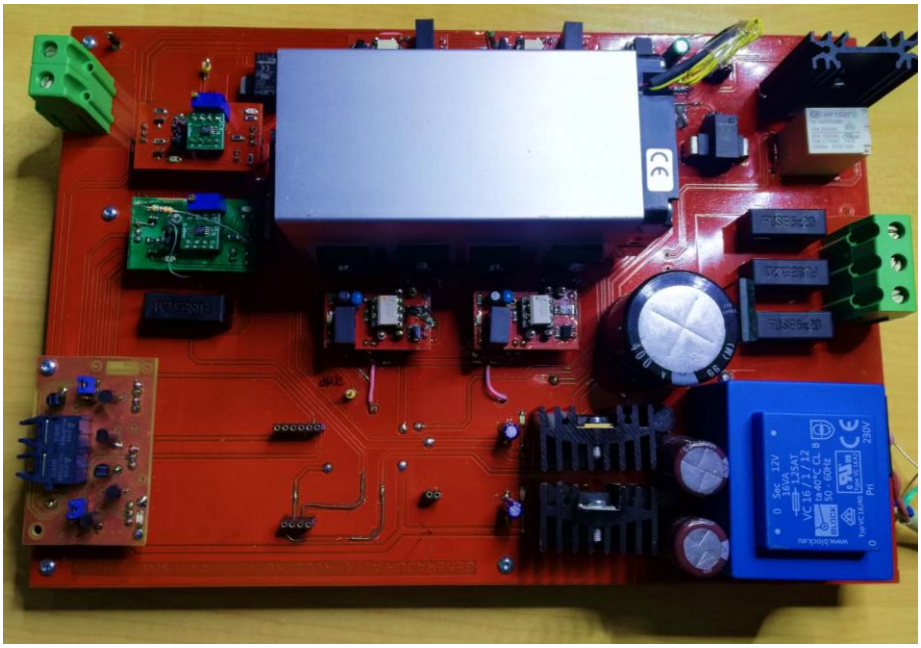


Ilustración 39 Puente inversor de onda completa. *Ira alojado dentro de un caja de apantallamiento.*

Hemos tomado medidas en un rango de frecuencias que va desde los 100kHz hasta 140kHz con un paso de 1,5kHz. En la ilustración 40 podemos ver la gráfica de la potencia aparente entregada por la red. Medida con la ayuda de un vatímetro.

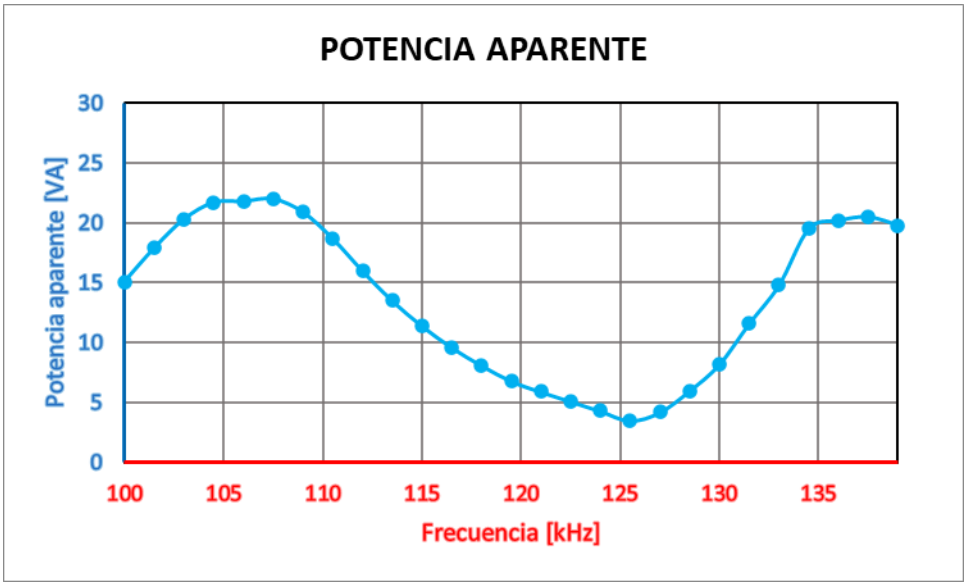


Ilustración 40 Potencia aparente frente a frecuencia

Hemos hallado experimentalmente la frecuencia de resonancia a 107,5kHz y a 137,5kHz. Mientras que la frecuencia de mínimo consumo la tendremos a 125,5kHz (recordemos que esta frecuencia era equivalente con la de oscilación natural del devanado secundario). Según nuestras estimaciones iniciales las frecuencias características del sistema deberían haber sido 109,645kHz, 120,235kHz y 139,780kHz. Como podemos observar no difieren mucho de los valores experimentales. Durante las pruebas pudimos ver que en la resonancia la tensión del bus se atenuaba bastante impidiendo que la potencia siga creciendo descontroladamente.

Destacamos que la potencia aparente que suministra la red. Terminará siendo la potencia activa que disipan las partes resistivas de la bobina. Durante un ciclo de recarga la red entrega potencia activa para alimentar las partes resistivas y potencia reactiva para recargar el condensador del inversor. La recarga del condensador se convertirá en potencia activa cuando la red deje de suministrar potencia hasta el siguiente ciclo. En la ilustración 41 podemos ver la forma que tiene la respuesta en frecuencia de la potencia activa y aparente que entrega el puente inversor a la bobina de Tesla. Si comparamos estas con la ilustración 40 observamos que está ultima tiene la forma de la potencia activa.

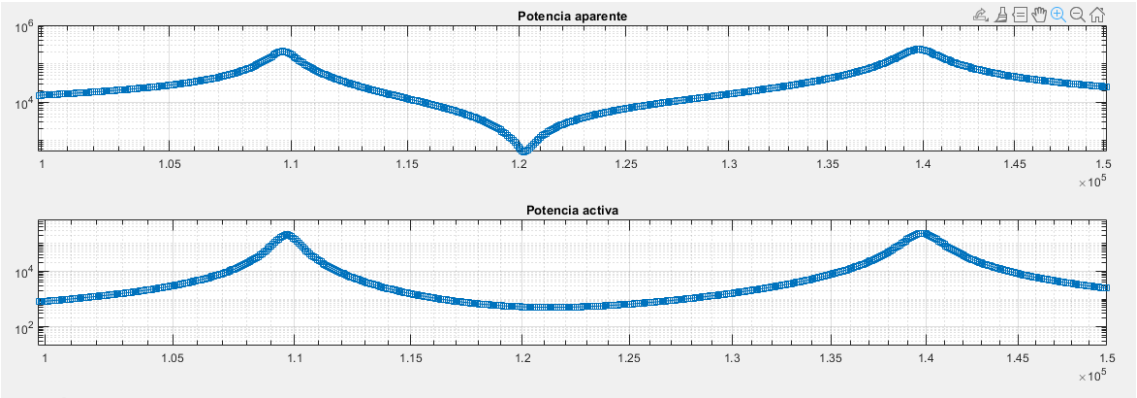


Ilustración 41 Tendencia de las potencias en una bobina de Tesla

Durante los ensayos también obtuvimos la respuesta del valor eficaz en un ciclo de red de la corriente de carga. Al alimentar a 15V la corriente no alcanza valores que puedan dañar nuestra placa. La respuesta de la corriente aparece en la ilustración 42.

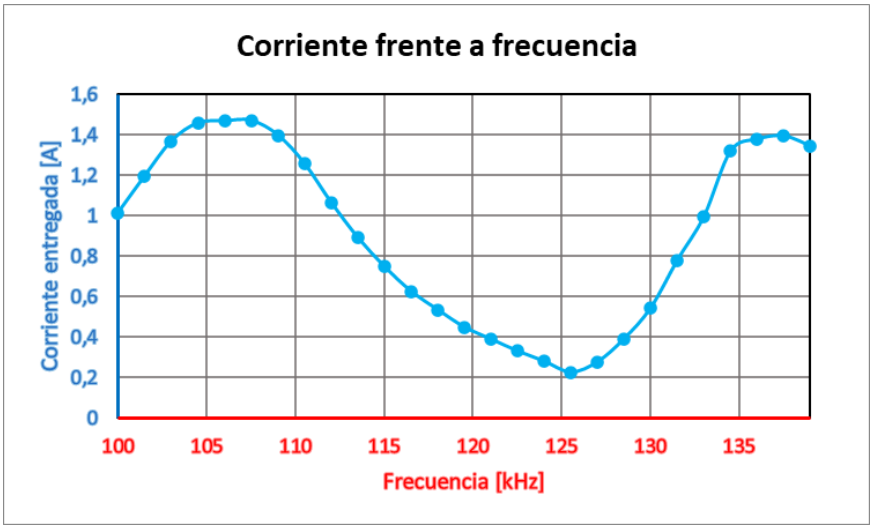


Ilustración 42 Corriente eficaz frente a frecuencia

La última parte de los ensayos consiste en observar cómo alternamos entre comportamiento capacitivo e inductivo conforme recorremos la curva de potencia. Para ello observaremos en la ilustración 43 el desfase entre el voltaje de bus y el voltaje del condensador primario respecto del esperado.

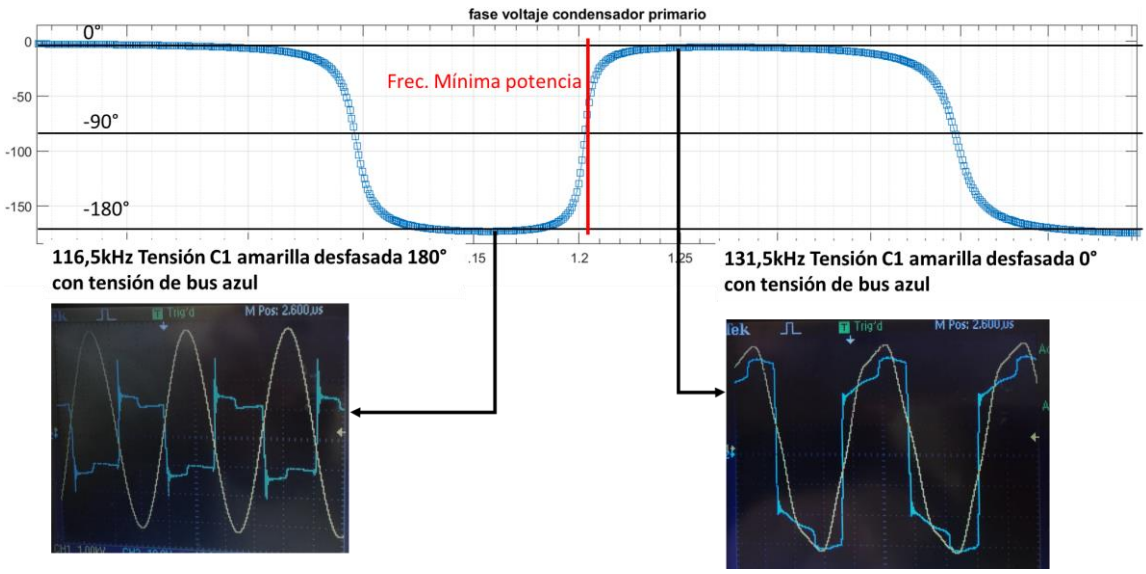


Ilustración 43 Alternancia del comportamiento con la frecuencia. *La pantalla izquierda está en la zona inductiva antes de la frecuencia de consumo mínimo. La de la derecha es después, en la zona capacitiva. La grafica superior sería la tendencia esperada.*

6. Conclusiones

Se ha construido un inversor de puente completo más robusto que su predecesor. Se ha logrado implementando las protecciones necesarias para que los elementos más importantes puedan trabajar siempre en unos rangos de funcionamiento seguros. Estas protecciones son contra las sobretemperaturas y sobrecorrientes. Diseñadas para proteger a los IGBTs siendo estos los elementos más importantes del puente inversor. A aparte de esto también se ha contado con otras protecciones pasivas para el resto de los elementos como puede ser el aislamiento galvánico presente en diversos componentes.

Fuera de las protecciones propias del puente de transistores. Destacamos el apantallamiento, esta protección era necesaria ya que la distorsión electromagnética generada por una bobina de Tesla afectará negativamente al funcionamiento de los componentes.

Como el punto más vulnerable a estas interferencias electromagnéticas es la conexión de la placa con el controlador. La unión entre ambas se realizó por un cable de fibra óptica. De esta forma se concluyó que la comunicación entre ambos módulos sería segura e inmune a las interferencias electromagnéticas de la bobina de Tesla. Habilitándonos la posibilidad de un control automático externo.

Gracias a la fibra se ha podido llevar el control fuera de la placa. A una distancia de trabajo segura para las personas y donde las interferencias electromagnéticas no afecten a la electrónica digital.

Se ha logrado el objetivo del control automatizado siendo este externo a la placa. El control se ejerce actuando sobre los impulsos de disparo de los IGBTs a partir de la información que nos retorna la propia placa. Esta nos enviará los datos más relevantes del funcionamiento usando como unidad de comunicaciones la UART de un Arduino Nano alojado dentro en la placa del inversor. Mientras que una unidad central, en este caso un Arduino Due, actuará sobre los pulsos de control de los IGBTs. El Arduino Due nos permite variar la frecuencia, dead time, ráfagas y duty cycle de los pulsos de disparo en los dos canales del inversor. Una ventaja añadida es que no estamos limitados a trabajar con un Arduino Due. La placa es compatible con cualquier control que tenga un puerto serie de lectura y sea capaz de generar una salida de frecuencia controlable.

La placa construida nos permite caracterizar en régimen permanente las curvas de comportamiento en frecuencia de una bobina de Tesla. El funcionamiento del inversor no estará limitado a la bobina de Tesla. Podrá ser utilizado en cualquier ámbito de aplicación compatible con sus funcionalidades y los límites de trabajo definidos.

7. Líneas futuras

La placa, aunque es operativa es claramente mejorable los puntos débiles sobre los que se debería actuar son:

- La protección de sobrecorrientes. Actualmente protege el sistema, pero no es precisa debido a los rebotes que se generan en su resistencia de shunt. Se debería poder implementar un control igual de rápido, pero más preciso.
- El sensado de señales. Actualmente esta la realiza el Arduino Nano a una velocidad de 10kS/s. Será necesario adaptar una lectura a má s velocidad si queremos medir el resto de los parámetros del puente inversor.

En lo que respecta a la respuesta en frecuencia. Los puertos de fibra instalados nos dan un gran abanico de posibilidades para el control de la placa y el monitoreo de señales. Tenemos una línea de fibra que nos permite controlar el inversor y otra línea que nos realimenta los datos de funcionamiento. Entre las posibilidades esta que el control y monitoreo podría ejecutarse por software desde un ordenador. Por ejemplo, utilizando Matlab podríamos graficar la respuesta en tiempo real e incluso obtener las gráficas de la respuesta en frecuencia automáticamente.

Respecto a la bobina de Tesla. El hecho poder controlar su comportamiento nos abre las puertas a otros proyectos. Como el de una bobina de Tesla musical o el de transmisión de energía inalámbrica.

Las posibilidades del puente inversor no acaban aquí. Podrá ser utilizado siempre que las condiciones de trabajo sean compatibles con los límites de funcionamiento definidos. Por ejemplo, en aplicaciones para el control de motores, o en convertidores DC-AC o DC-DC.

8. Bibliografía

[1] Varios, «Wikipedia,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/Bobina_de_Tesla. [Último acceso: 18 9 2021].

[2] F. Zapata, «Lifeder,» 6 4 2020. [En línea]. Available: https://www.lifeder.com/bobina-de-tesla/. [Último acceso: 22 9 2021].

[3] A. Cahun, «Xataka,» 19 Octubre 2020. [En línea]. Available: https://www.xataka.com.mx/celulares-y-smartphones/xiaomi-presenta-carga-inalambrica-rapida-mundo-80w-para-cargar-cables-smartphone-4-000-mah-19-minutos. [Último acceso: 19 9 2021].

[4] E. Pérez, «Xataka,» 6 Noviembre 2020. [En línea]. Available: https://www.xataka.com/vehiculos/recarga-inalambrica-para-coches-electricos-da-importante-salto-nuevo-estandar-alcanza-11-kw-25-centimetros-distancia. [Último acceso: 19 9 2021].

[5] J. Gómara, «Hibridos y eléctricos,» 10 Agosto 2021. [En línea]. Available: https://www.hibridosyelectricos.com/articulo/tecnologia/recarga-inalambrica-1-mw-tesla-semi/20210810111953047824.html. [Último acceso: 21 9 2021].

[6] D. Zuriaga Miguel, «Diseño, construcción y ensayos del circuito excitador de una bobina de tesla de estado sólido (SSTC),» Repositorio Zaguan, Zaragoza, 2018.

[7] S. Menjíbar Ruiz, «Diseño paramétrico de bobinas Tesla,» Repositorio Zaguan, Zaragoza, 2017.

[8] S. Menjibar Ruiz, «Anexo 4 sección 3,» de *Diseño paramétrico de bobinas de Tesla*, Zaragoza, Repositorio Zaguan, 2017, pp. 62-63.

[9] L. Barrierè, «Departamento de matematicas-Universitat politécnica de catalunya,» Octubre 2011. [En línea]. Available: https://web.mat.upc.edu/lali.barriere/as/serie-fourier.pdf. [Último acceso: 5 10 2021].

[10] IXYS, «FUO50-16N,» 2019. [En línea]. Available: https://www.mouser.es/datasheet/2/240/FUO50-16N-1549180.pdf. [Último acceso: 22 10 2021].

[11] UNAD, «Universidad nacional abierta y a distancia. Eléctronica industrial,» UNAD, [En línea]. Available: https://docplayer.es/37170453-Rectificador-trifasico-de-onda-completa.html. [Último acceso: 22 10 2021].

[12] N. Mohan, T. M. Undeland y W. P. Robbins, «27-5 Amortiguador de apagado,» de *Electronica de potencia convertidores aplicaciones y diseño*, Mc Graw Hill, 2009, p. 596.

[13] ON Semiconductor, «IGBT FGH40N60SMDF,» ON Semiconductor, [En línea]. Available: https://www.mouser.es/datasheet/2/308/1/FGH40N60SMDF_D-2313301.pdf. [Último acceso: 20 10 2021].

[14] Linear Technology, «DATASHEET LT1711,» Linear technology, [En línea]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/171112f.pdf. [Último acceso: 20 10 2021].

[15] T. TLP250H, «Toshiba Electronic Devices & Storage Corporation,» TOSHIBA, 24 12 2015. [En línea]. Available: https://toshiba.semicon-storage.com/eu/semiconductor/product/optoelectronics/detail.TLP250HF.html. [Último acceso: 20 11 2021].

[16] LEM, «HLSR-P SERIES,» LEM, [En línea]. Available: https://www.lem.com/sites/default/files/products_datasheets/hlsr-p_series.pdf. [Último acceso: 1 11 2021].

[17] ST Electronics, «SG3525 REGULATING PULSE WIDTH MODULATORS,» ST Electronics, [En línea]. Available: https://docs.rs-online.com/374e/0900766b8140da7e.pdf. [Último acceso: 21 10 2021].

9. Índice de ilustraciones

ANEXOS

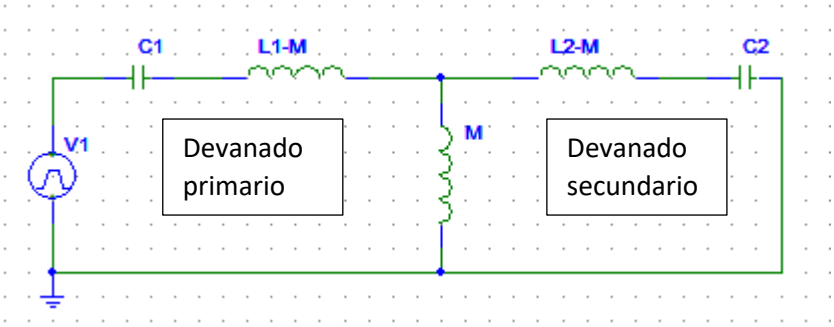
Anexo I: Cálculos

En este anexo repasaremos los siguientes estudios:

- Análisis por transformada de Laplace de las frecuencias de resonancia
- Cálculo de las corrientes por condensador del rectificador
- Análisis de los modos de conmutación del IGBT
- Análisis de las potencias térmicas disipadas

Frecuencias de resonancia de una bobina de Tesla

Se va a analizar el siguiente circuito que también aparece en la ilustración 6 de la página 5. Vamos a ver cómo obtener las frecuencias de resonancia de una bobina de Tesla.



El desarrollo de los cálculos ha sido extraído de la referencia [8]. Las ecuaciones que modelan este circuito son:

$$\left(L_1 \cdot s + \frac{1}{C_1 \cdot s}\right) \cdot I_1 + M \cdot s \cdot I_2 = \frac{b_1 \cdot \omega_0}{s^2 \cdot {\omega_0}^2} \text{ (Excitación senoidal de frecuencia } \omega_0)$$

$$M \cdot s \cdot I_1 + \left(L_2 \cdot s + \frac{1}{C_2 \cdot s}\right) \cdot I_2 = 0$$

Donde el termino I_1 es la corriente por devanado primario y I_2 la del devanado secundario. El termino donde aparece la frecuencia de excitación es la transformada de Laplace de la tensión senoidal. Estamos analizando la función de transferencia entre la tensión de alimentación y la corriente del devanado primario. El siguiente paso será despejar I_1:

$$I_1 = -\frac{1}{M \cdot s} \cdot \left(L_2 \cdot s + \frac{1}{C_2 \cdot s}\right) \cdot I_2$$

$$-\frac{1}{M \cdot s} \cdot \left(L_2 \cdot s + \frac{1}{C_2 \cdot s}\right) \cdot I_2 \cdot \left(L_1 \cdot s + \frac{1}{C_1 \cdot s}\right) + M \cdot s \cdot I_2 = \frac{b_1 \cdot \omega_0}{s^2 \cdot {\omega_0}^2}$$

$$\left(-(1 + L_1 \cdot C_1 \cdot s^2) \cdot (1 + L_2 \cdot C_2 \cdot s^2) + (M^2 \cdot C_1 \cdot C_2 \cdot s^4)\right) \cdot I_2 = \frac{b_1 \cdot M \cdot \omega_0 \cdot C_1 \cdot C_2 \cdot s^3}{s^2 + {\omega_0}^2}$$

Las raíces del polinomio izquierdo serán los polos del sistema. Por lo que el polinomio característico será:

$$(L_1 \cdot C_1 \cdot L_2 \cdot C_2 - C_1 \cdot C_2 \cdot M^2) \cdot s^4 + (L_1 \cdot C_1 + L_2 \cdot C_2) \cdot s^2 + 1 = 0$$

Las raíces de este polinomio son:

$$x^2_{1,2} = \frac{-(L_1 \cdot C_1 + L_2 \cdot C_2) \pm \sqrt{(L_1 \cdot C_1 + L_2 \cdot C_2)^2 - 4 \cdot (L_1 \cdot C_1 \cdot L_2 \cdot C_2 - C_1 \cdot C_2 \cdot M^2)}}{2 \cdot (L_1 \cdot C_1 \cdot L_2 \cdot C_2 - C_1 \cdot C_2 \cdot M^2)}$$

Podemos utilizar las siguientes igualdades para despejar términos en las raíces:

$$k^2 = \frac{M^2}{L_1 \cdot L_2} \quad ; \quad \omega_1{}^2 = \frac{1}{L_1 \cdot C_1} \quad ; \quad \omega_2{}^2 = \frac{1}{L_2 \cdot C_2}$$

Donde k es el coeficiente de acoplamiento y ω la frecuencia natural del devanado correspondiente. Despejando en la ecuación anterior obtendremos:

$$x^2_{1,2} = \frac{-(\omega_1{}^2 + \omega_2{}^2) \pm \sqrt{(\omega_1{}^2 + \omega_2{}^2)^2 - 4 \cdot ((1-k^2) \cdot \omega_1{}^2 \cdot \omega_2{}^2)}}{2 \cdot (1-k^2)}$$

Utilizaremos el siguiente cambio de variable para facilitar la resolución:

$$\sigma^2 = 1 - \frac{M^2}{L_1 \cdot L_2} = 1 - k^2$$

Entonces las raíces serán:

$$x^2_{1,2} = \frac{-(\omega_1{}^2 + \omega_2{}^2)}{2 \cdot \sigma^2} \pm \sqrt{\frac{(\omega_1{}^2 + \omega_2{}^2)^2}{2 \cdot \sigma^2} - \frac{\omega_1{}^2 \cdot \omega_2{}^2}{\sigma^2}}$$

Por último, si consideramos que las frecuencias naturales de ambos sistemas son idénticas los polos del sistema nos quedarán de la siguiente forma:

$$x^2_{1,2} = \frac{-\omega_1{}^2}{\sigma^2} \pm \sqrt{\frac{\omega_1{}^4}{\sigma^4} - \frac{\omega_1{}^2 \cdot \omega_2{}^2}{\sigma^2}} = \frac{\omega_1{}^2}{\sigma^2} \pm \sqrt{\frac{\omega_1{}^4}{\sigma^4} \cdot (1-\sigma^2)} = \frac{\omega_1{}^2}{\sigma^2} \pm \frac{\omega_1{}^2}{\sigma^2} \cdot \sqrt{(1-\sigma^2)}$$

$$x^2_{1,2} = \omega_1{}^2 \cdot \frac{1 \pm \sqrt{(1-\sigma^2)}}{\sigma^2} = \omega_1{}^2 \cdot \frac{1 \pm k}{1-k^2} = \omega_1{}^2 \cdot \frac{(1 \pm k)}{(1+k)(1-k)}$$

$$x^2_1 = \frac{\omega_1{}^2}{1-k}$$

$$x^2_2 = \frac{\omega_1{}^2}{1+k}$$

Finalmente, si pasamos estos valores a hercios nos quedan las frecuencias de resonancia de las ecuaciones 4 y 5 de la página 6:

$$f_1 = \frac{x_1}{2 \cdot \pi} = \frac{\sqrt{\frac{\omega_1{}^2}{1-k}}}{2 \cdot \pi}$$

$$f_2 = \frac{x_2}{2 \cdot \pi} = \frac{\sqrt{\frac{\omega_1{}^2}{1+k}}}{2 \cdot \pi}$$

Corrientes de carga por el condensador rectificador

Se va a estudiar los valores de corriente que circulan por el condensador rectificador. Sabemos que por un condensador la corriente tiene la siguiente forma:

$$I = C\frac{dV}{dt}$$

Durante su recarga la tensión del condensador será la tensión que tiene la red en ese momento. Se ha fijado un rizado máximo del 15% por lo que la tensión que carga del condensador será:

$$V = 325 \cdot \sin(2\pi \cdot 50 \cdot t)$$

La recarga será entre los instantes en los que el término senoidal vaya desde un valor de 0,85 hasta 1. Cada ciclo de red tendrá dos recargas para la configuración monofásica y seis en el caso de la trifásica. Se considerará que la corriente por el condensador tendrá la siguiente forma:

$$I = C \cdot 325 \cdot 2\pi \cdot 50 \cdot \cos(2\pi \cdot 50 \cdot t)$$

Esta no deja de ser una aproximación, la forma de la corriente real se asemejará más a un pulso cuadrado que a un coseno. Según la ecuación anterior la corriente será máxima para el instante en el que se inicie la recarga. Cuando el valor de la tensión sea 85% sobre su amplitud total. Sabiendo que la capacidad es de 470[µF] la corriente máxima queda:

$$Imax = 470 \cdot 10^{-6} \cdot 325 \cdot 2\pi \cdot 50 \cdot \cos(arcsen(0{,}85)) = 25{,}28[\text{A}]$$

Una vez obtenida la corriente máxima pasaremos a calcular la corriente eficaz en un ciclo de red. La corriente eficaz sigue la siguiente ecuación para la configuración monofásica:

$$Ieff^2 = \frac{2}{Tred}\int_{arcsen(0{,}85)}^{arcsen(1)} I^2 d\omega t =$$

$$Ieff^2 = \frac{2}{0.02}\int_{arcsen(0{,}85)}^{arcsen(1)} (470 \cdot 10^{-6} \cdot 325 \cdot 2 \cdot \pi \cdot 50)^2 \cdot \frac{COS(\omega t)^2}{\omega} d\omega t = 2 \cdot 19{,}6165$$

Hay que tener en cuenta que la carga ocurre dos veces en el caso de la monofásica y 6 en la trifásica. Sabiendo esto tenemos que las corrientes eficaces son:

$$Ieff\ monofásica = \sqrt{2 \cdot 19{,}6165} = 6{,}26[A]$$

$$Ieff\ trifásica = \sqrt{6 \cdot 19{,}6165} = 10{,}85[A]$$

Potencia térmica disipada

En este apartado analizaremos el calor que disipan los IGBTs del puente inversor. Considerando para ello el peor de los casos. Cuando el comportamiento sea fuertemente inductivo, con una forma de corriente triangular. Para estos cálculos vamos a considerar una corriente máxima en la conmutación de 20A en el transistor que al ser triangular su valor eficaz será 11,5A. Según las consideraciones iniciales las pérdidas en conmutación de un transistor responden a la siguiente ecuación:

$$Pswitch = \frac{V \cdot I \cdot tcoff}{2} \cdot fmax = \frac{325 \cdot 20 \cdot 20 \cdot 10^{-9}}{2} \cdot 200000 = 13[W]$$

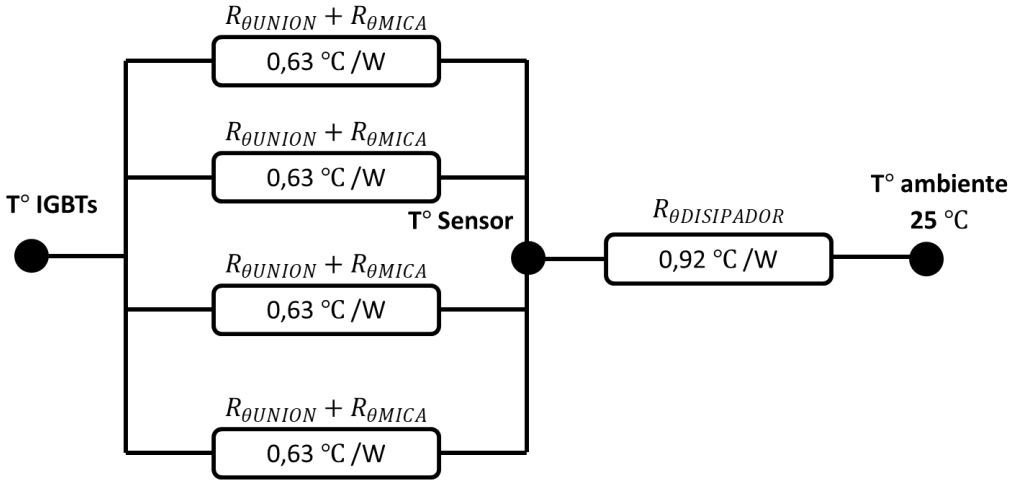
Al estar en un funcionamiento inductivo parte del ciclo estará conduciendo el transistor y la otra parte el diodo de conducción inversa. Con la finalidad de simplificar los cálculos vamos a considerar que ambos tienen las mismas perdidas en conducción. Las perdidas en conducción de un solo transistor son

$$Pcond = 0{,}5 \cdot Rcond \cdot Ieff^2 = 0{,}5 \cdot 0{,}0625 \cdot 11{,}5^2 = 4{,}13[W]$$

Por tanto, la potencia que disipan los 4 IGBTs será:

$$Ptotal = 4 \cdot (Pswitch + Pcond) = 68{,}53[W]$$

Esta es la potencia disipada a 200[kHz]. Introduciendo este parámetro en el siguiente esquema de resistencias térmicas podemos estimar la temperatura de los IGBTs



Sabiendo que el equivalente térmico de las resistencias de los IGBTs es de la forma:

$$R_{\theta equiv.} = (R_{\theta\ unión+mica} // R_{\theta\ unión+mica} // R_{\theta\ unión+mica} // R_{\theta\ unión+mica})$$

Con un valor de 0,1575 [°C/W]. La temperatura que alcanzará el IGBT es la siguiente:

$$T°igbt = Ptotal \cdot (0{,}1575 + 0{,}92) + 25 = 98{,}84\ [°C]$$

Mientras que la temperatura en el sensor cuando la del IGBT sea de 110[°C] se calculará como un divisor de temperatura:

$$T°sensor = \frac{110-25}{0{,}92+0{,}1575} \cdot 0{,}92 + 25 = 97{,}6[°C]$$

Anexo II: Firmware

En este anexo veremos todos los programas en el proyecto. Diferenciaremos 3 tipos de programas. Entre los programas de Matlab, Arduino Nano y Arduino Due.

Programa de Matlab

Solo utilizaremos un código de Matlab. El que hemos utilizado para los diagramas de Bode.

```
%Programa escrito y diseñado por Julio Sebastián Sullca Trillo
%para el proyecto Diseño y construcción de una placa electrónica
%para el control de una bobina Tesla

clear all

%bobina tesla
syms W
syms S
C1 =75e-9 ;%faradios
C2 =21e-12;%faradios
     L1 = 22.015e-6;%henrios
     L2 = 83.4e-3;%henrios
R1 = 284.74e-3;%ohmios
R2 =352;%ohmios
M  = 0.32e-3;%henrios

ZC1 =-j/(C1*W) ;%faradios
ZC2 =-j/(C2*W);%faradios
ZL1 = L1*W*j;%henrios
ZL2 = L2*W*j;%henrios
ZR1 = R1;%ohmios
ZR2 =R2;%ohmios
ZM  = M*W*j;%henrios

SC1 =1/(C1*S) ;%faradios
SC2 =1/(C2*S);%faradios
SL1 = L1*S;%henrios
SL2 = L2*S;%henrios
SR1 = R1;%ohmios
SR2 =R2;%ohmios
SM  = M*S;%henrios

%Una vez introducidos los parametros de la bobina
%calculamos los parámetros electricos

V=230*sqrt(2)*4/(pi);%Primer armonico

% Calculamos la impedancia equivalente
Zeq=(ZR1+ZC1+ZL1-ZM)+(ZM*(ZL2-ZM+ZR2+ZC2)/(ZM+ZL2-ZM+ZR2+ZC2));

%figure(1)
%GRAFICA DE LA IMPEDANCIA
x1 = logspace(4.69,5.30103,2000);
frec1=2*pi*x1;
g1=subs(Zeq,W,frec1);
y1=abs(g1);
t=angle(g1)*180/pi;

figure(1)
tiledlayout(2,1)
```

```
nexttile
loglog(x1,y1,'-s');
title('impedancia equivalente')
grid on
nexttile
semilogx(x1,t,'-s');
title('fase impedancia equivalente')
grid on

%figure (2)
%CORRIENTE TOTAL
Ic2=(V-((ZR1+ZC1+ZL1-ZM)*V/Zeq))/(ZL2-ZM+ZR2+ZC2);%corriente por
   condenasador secundario, lo usaremos mas adelante
I=V/Zeq;%corriente total
x2 = logspace(4.69,5.30103,5000);
frec2=2*pi*x2;
g2=subs(I,W,frec2);
y2=abs(g2);
t2=angle(g2)*180/pi;

figure(2)
tiledlayout(2,1)
nexttile
loglog(x2,y2,'-s');
title('corriente total')
grid on
nexttile
semilogx(x2,t2,'-s');
title('fase corriente total')
grid on

%figure (3)
%VOLTAJE CONDENSADOR SECUNDARIO
Vc2=ZC2*Ic2;

x3 = logspace(4.69,5.30103,2000);
frec3=2*pi*x3;
g3=subs(Vc2,W,frec3);
y3=abs(g3);
t3=angle(g3)*180/pi;

figure(3)
tiledlayout(2,1)
nexttile
loglog(x3,y3,'-s');
title('voltaje condensador')
grid on
nexttile
semilogx(x3,t3,'-s');
title('fase voltaje condensador secundario')
grid on

%figure (4)
%POTENCIAS
S=V*I*0.5;%valor eficaz
x4 = logspace(4.69,5.30103,2000);
frec4=2*pi*x4;
g4=subs(S,W,frec4);
y4=abs(g4);%potencia aparente
t4=angle(g4);
p4=real(g4);
```

```
q4=-imag(g4);%aparente negativa es capacitiva
%fdp
fdp=cos(t4);
figure(4)
tiledlayout(3,1)
nexttile
loglog(x4,y4,'-s');
title('Potencia aparente')
grid on
nexttile
loglog(x4,p4,'-s');
title('Potencia activa')
grid on
nexttile
semilogx(x4,q4,'-s');
title('Potencia reactiva')
grid on


%figure (5)
%VOLTAJE CONDENSADOR PRIMARIO

Vc1=ZC1*I;
x5 = logspace(4.69,5.30103,2000);
frec5=2*pi*x5;
g5=subs(Vc1,W,frec5);
y5=abs(g5);
t5=angle(g5)*180/pi;

figure(5)
tiledlayout(2,1)
nexttile
loglog(x5,y5,'-s');
title('voltaje condensador primario')
grid on
nexttile
semilogx(x5,t5,'-s');
title('fase voltaje condensador primario')
grid on

figure(6)%FACTOR DE POTENCIA


semilogx(x4,fdp,'-s');
title('Factor de potencia')
grid on
```

Programa del Arduino Nano

Este es el programa encargado de ejecutar el sensado de variables sobre la placa y generar una retroalimentación al procesador externo. Para la comunicación con la pantalla LCD será necesario activar la biblioteca indicada en el programa.

```
1. //Programa escrito y diseñado por Julio Sebastián Sullca Trillo
   para el proyecto Diseño y construcción de una placa electrónica
   para el control de una bobina Tesla
2.
3. //------------------DEFINES PARA LAS CONVERSIONES---------------
4. #define I_pin A2 //pin para la lectura de la corriente
5. #define V_pin A0 //pin para la lectura del voltaje
```

```
#define Temp_pin A1 //pin para la lectura de la temperatura
//-------------------------DEFINES PARA LOS CALCULOS-----------
#define Puerto_I_ENT 0 //NOS SITUA EN EL VECTOR DE VALORES PARA LEER LA CORRIENTE
#define Puerto_V_LIN 150 //NOS SITUA EN EL VECTOR DE VALORES PARA LEER EL VOLTAJE
#define frec 100 // frecuencia de la red rectificada
//----------------------DEFINES PARA LAS COMUNICACIONES----
// Para el control de interrupciones y comunicaciones sera necesario las siguientes variables
#define intComp1 2 //PIN DE INTERRUPCION DE ARDUINO INT0
#define intComp2 3 //PIN DE INTERRUPCION DE ARDUINO INT1
#define control_pin 6 //PIN PARA DETENER LOS PULSOS
#define latch_pin1 5 //PIN PARA DESACTIVAR EL LATCH COMP1
#define latch_pin2 9 //PIN PARA DESACTIVAR EL LATCH COMP1
#define T_max 97,6 //temperatura maxima
#define T_histeresis 75 //temperatura minima
void comunicaciones (void);

//------------------------biblioteca de la pantalla--------
#include <LCD_I2C.h>

LCD_I2C lcd(0x3F); // direccion de la pantalla

//------------------------VARIABLES DEL ADC----------------

volatile int valores[301] ; // del 0 al 149 voltaje, del 149 a 299 corriente, en el 300 temperatura
volatile int i = 0; // indica cuantas conversiones llevamos en cada etapa

void borrar_vectores (void);
void lectura_adc (void);

//----------------------VARIABLES PARA LOS CALCULOS--------

int I_entrada_max; //ALMACENA EL VALOR MAXIMO DE CORRIENTE ENTREGADA
int V_max; //ALMACENA EL VALOR MAXIMO DE TENSION DEL BUS
int V_min; //ALMACENA EL VALOR MINIMO DE TENSION DEL BUS
int t_on; //ALMACENA EL INSTANTE DISCRETO EN EL QUE LA CORRIENTE EMPIEZA A CIRCULAR
int t_off; //ALMACENA EL INSTANTE DISCRETO EN EL QUE LA CORRIENTE DEJA DE CIRCULAR
long suma_I; //LA SUMA DE LAS CORRIENTES POSITIVAS PARA EL CALCULO DE LA POTENCIA
float V_medio;//VOLTAJE MEDIO DE LA ONDA TRIANGULAR APROXIMADA
float I_carga;//CORRIENTE DE CARGA MEDIA
float temperatura;//TEMPERATURA DEL LM35
int k ; //VARIABLE PARA RECORRER EL VECTOR DE VALORES
long P; //ALMACENA EL VALOR DE LA POTENCIA ACTIVA ENTREGADA
int vueltas_contadas = 0; //cada 15 bucles actualizaremos el valor de la pantalla LCD


//--------------VARIABLES PARA ESCRIBIR POR PANTALLA LCD---
int Vinteger;
int Iinteger;
int Tempinteger;
int Pinteger;

```

```
57. void calculos (void);
58. //--------VARIABLES DE LA RUTINA DE INTERRUPCION EXTERNA---
59. volatile int Flag = 0;
60. volatile int interrupcion = 0;
61. void rutina_excepcion1(void);
62. void rutina_excepcion2(void);
63.
64.
65. //-------------------------------------SET UP-----------
66. void setup() {
67.   cli();
68.   Serial.begin(115200);
69.
70.   //CONFIGURACION DE LOS PINES EXTERNOS OJO A COMO ESTAN
    CONECTADOS LOS TRANSISTORES
71.
72.   //INTERRUPCION POR MALFUNCIONAMIENTO COMPARADOR 1
73.   pinMode(intComp1, INPUT);//PIN DE LA INTERRUPCION EXTERNA
74.   attachInterrupt(digitalPinToInterrupt(intComp1),
    rutina_excepcion1, FALLING);//CONDICION DE INTERRUPCION
75.   //INTERRUPCION POR MALFUNCIONAMIENTO COMPARADOR 2
76.   pinMode(intComp2, INPUT);//PIN DE LA INTERRUPCION EXTERNA
77.   attachInterrupt(digitalPinToInterrupt(intComp2),
    rutina_excepcion2, FALLING);//CONDICION DE INTERRUPCION
78.
79.   //SALIDAS Y ENTRADAS DIGITALES
80.   pinMode(latch_pin1, OUTPUT); //PIN PARA ELIMINAR LATCH COMP 1
81.   digitalWrite(latch_pin1, HIGH); //LATCH LISTO PARA FUNCIONAR
82.   pinMode(latch_pin2, OUTPUT); //PIN PARA ELIMINAR LATCH COMP 2
83.   digitalWrite(latch_pin2, HIGH); //LATCH LISTO PARA FUNCIONAR
84.   pinMode(control_pin, OUTPUT); //PIN QUE ASEGURA LA ANULACION
    DE LOS PULSOS DURANTE LA EXCEPCION
85.   digitalWrite(control_pin, HIGH); //ASEGURA QUE LOS PULSOS SE
    PUEDAN PRODUCIR
86.   // Habilite todas las interrupciones.
87.   sei();
88.
89.   lcd.begin(); // inicializamos la pnatalla LCD
90.   lcd.backlight();
91. }
92.
93. void loop() {
94.
95.   lectura_adc();
96.   calculos();
97.   comunicaciones();
98.
99. }
100.
101.
102. //-------------FUNCION PARA LAS 300 LECTURAS DEL ADC-------
103. void lectura_adc (void) {
104.
105.   //INICIALIZAMOS LAS VARIABLES DE CONTROL
106.   borrar_vectores();//Resetea el vector que guarda las lecturas
107.   i = 0; //Toma valores del 0 al 300, siendo el 100 una
    referencia para finalizar las lecturas
108.
109.   //BUCLE DE LECTURA
110.   while ((i < 150) && (Flag == 0)) {
111.     valores[i] = analogRead(I_pin); //leemos el voltaje del bus
```

```
112.     i++;
113.   }
114.   //BUCLE DE LECTURA
115.   while ((i < 300) && (Flag == 0)) {
116.     valores[i] = analogRead(V_pin); //leemos La corriente de
   entrada
117.     i++;
118.   }
119.   //BUCLE DE LECTURA
120.   valores[300] = analogRead(Temp_pin); // leemos la temperatura
121.
122.
123. }
124.
125. //-----------------------FUNCION DE LOS CALCULOS----------
126.
127. void calculos(void) {
128.
129.   //REINICIAR LAS VARIABLES DE LAS OPERACIONES
130.   t_on = 0; //instante en el que la corriente supera el 25% del
   valor medio
131.   t_off = 0; //instante en el que la corriente baja del 25% del
   valor medio
132.
133.   suma_I = 0; //suma de las corrientes positivas
134.   k = 0; //variable para recorrer el vector de valores
135.
136.   I_entrada_max = 0; //VALOR I MAXIMO DE LA ENTRADA
137.   V_max = 0; //VALOR V MAXIMO
138.   V_min = 1023; //VALOR V MAXIMO
139.
140.   V_medio = 0.0; //TENSION MEDIA
141.   I_carga = 0.0; //CORRIENTE DE CARGA
142.
143.   P = 0; //potencia
144.
145.
146.   //Hacemos operaciones con los vectores
147.   for (int h = 0; h < 150; h++) {
148.     //Rompe el bucle for en caso de interrupcion
149.     if (Flag == 1) {
150.       break;
151.     }
152.     //Buscamos la corriente de entrada maxima
153.     if ((valores[h + Puerto_I_ENT]) > I_entrada_max) {
154.       I_entrada_max = valores[h + Puerto_I_ENT];
155.     }
156.
157.     //Buscamos el voltaje maximo
158.     if ((valores[h + Puerto_V_LIN]) > V_max) {
159.       V_max = valores[h + Puerto_V_LIN];
160.     }
161.
162.     //Buscamos el voltaje minimo
163.     if ((valores[h + Puerto_V_LIN]) < V_min) {
164.       V_min = valores[h + Puerto_V_LIN];
165.     }
166.
167.   }
168.
```

```
169.   V_medio = (V_max + V_min) / 2.0; //Calculamos el valor medio
   suponemos onda triangular de tension
170.
171.   V_medio = (5.0 / 1024.0) * V_medio * (68820.0 / 820.0); //CALC
   ULAMOS LA TENSION MEDIA DEL BUS DE CONTINUA
172.
173.   while ((valores[k + Puerto_I_ENT] > (I_entrada_max * 0.25)) &&
    (k < 150)) { //Nos da el instante EN QUE TERMINA UN PULSO SI ESTE
   NO HA ACABADO DEL VECTOR ANTERIOR
174.     k++;
175.     //Rompe el bucle for en caso de interrupcion
176.     if (Flag == 1) {
177.       break;
178.     }
179.   }
180.
181.   while ((valores[k + Puerto_I_ENT] < (I_entrada_max * 0.25)) &&
    (k < 150)) { //Nos da el instante de Ton
182.     k++;
183.     //Rompe el bucle for en caso de interrupcion
184.     if (Flag == 1) {
185.       break;
186.     }
187.   }
188.   t_on = k; //instante en el que la corriente sube
189.
190.
191.   while ((valores[k + Puerto_I_ENT] >= (I_entrada_max * 0.25)) &
   & (k < 150)) { //Nos da el instante de Toff
192.     suma_I = suma_I + valores[k + Puerto_I_ENT];
193.     k++;
194.     //Rompe el bucle for en caso de interrupcion
195.     if (Flag == 1) {
196.       break;
197.     }
198.   }
199.   t_off = k; //instante en el que la corriente ha bajado
200.
201.   I_carga = I_entrada_max; //corriente de carga sin convertir
202.   I_carga = (((5.0 / 1024.0) * I_carga) - 2.5) * (50.0 / 2.0); /
   /referencia a 2,5V la conversion del hall es 50/2
203.
204.   P = (0.0001 * (t_off - t_on) * (V_medio * I_carga) * (frec)) +
    0.5; //redondea de float a long
205.
206.   temperatura = valores[300] * (5.0 / 1024.0) * 100.0; //10mv/ยบ
   c del LM35
207.
208.   //redondeamos los flotantes a enteros
209.   vueltas_contadas++;
210.   if (vueltas_contadas == 10) {
211.     Vinteger = (int)V_medio;
212.     Iinteger = (int)I_carga;
213.     Tempinteger = (int)temperatura;
214.     Pinteger = (int)P;
215.     vueltas_contadas = 0;
216.   }
217. }
218. //--------------------FUNCION PARA COMUNICACIONES---------
219. void comunicaciones (void) {
```

```
220.   lcd.clear();//Va ha empezar una nueva comunicacion
221.   if (temperatura > T_max) {
222.     Flag = 1;
223.     interrupcion = 3; //temperatura al limite
224.   }
225.   if (Flag == 1) {
226.     digitalWrite(control_pin, LOW);//cerramos los pulsos
227.     Serial.print(F("N\r\n"));//mensaje de error
228.     Serial.println(interrupcion);
229.     lcd.print("ERROR:");//escribimos el motivo del error por la
   pantalla
230.     lcd.setCursor(8, 0);
231.     lcd.print(interrupcion);
232.     //delay(3000);//NECESARIO PARA BANCO DE PRUEBAS
233.     if ((interrupcion == 1) || (interrupcion == 2)) { //error
   por sobrecorrientes
234.       while (((digitalRead(intComp1)) == 0) || ((digitalRead(int
   Comp2)) == 0)) {
235.
236.         digitalWrite(latch_pin1, LOW);//Ya sabemos que
   comparador ha saltado deslatcheamos
237.         digitalWrite(latch_pin2, LOW);//Ya sabemos que
   comparador ha saltado deslatcheamos
238.
239.       }
240.
241.       digitalWrite(latch_pin1, HIGH);// hemos deslatcheado,
   activamos el pin de latch para la siguiente iteracion
242.       digitalWrite(latch_pin2, HIGH);// hemos deslatcheado,
   activamos el pin de latch para la siguiente iteracion
243.       delay(200);
244.     }
245.     else {
246.       while (temperatura > T_histeresis) { //esperamos a que la
   temperatura baje lo suficiente
247.         delay(200);
248.       }
249.     }
250.     Flag = 0;//reseteamos la alerta
251.     interrupcion = 0;//reseteamos la interrupcion
252.     digitalWrite(control_pin, HIGH);//lanzamos los pulsos de
   nuevo
253.     Serial.print(F("F"));//el estado de fallo ha terminado
254.   }
255.   else {
256.
257.     Serial.print(F("Y\r\n"));//mensaje de todo en orden
258.     Serial.print(I_carga); Serial.print(F("\r\n"));
259.     Serial.print(V_medio); Serial.print(F("\r\n"));
260.     Serial.print(P); //Serial.print(F("\r\n"));
261.     Serial.print(F("F"));//se termino el mensaje
262.     //escribimos por la pantalla
263.     lcd.print("V: ");//escribimos el valor de la tension de bus
264.     lcd.setCursor(3, 0);
265.     lcd.print(Vinteger);//escribimos el valor de la tension de
   bus
266.     lcd.setCursor(8, 0);
267.     lcd.print("I: ");//escribimos el valor de la corriente
268.     lcd.setCursor(12, 0);
269.     lcd.print(Iinteger);//escribimos el valor de la tension de
   bus
```

```
270.     lcd.setCursor(0, 1);
271.     lcd.print("P: ");//escribimos el valor de la potencia
272.     lcd.setCursor(3, 1);
273.     lcd.print(Pinteger);//escribimos el valor de la potencia
274.     lcd.setCursor(8, 1);
275.     lcd.print("T: ");//escribimos la temperatura
276.     lcd.setCursor(12, 1);
277.     lcd.print(Tempinteger);//escribimos el valor de la
   temperatura
278.     delay(50);
279.
280.   }
281.
282. }
283.
284. //-----------------FUNCION PARA BORRAR VECTORES-----------
285. void borrar_vectores (void) {
286.   for (int q = 0; q < 301 ; q++) {
287.     valores[q] = 0; //Pone a 0 todos los valores del vector de
   lecturas
288.   }
289. }
290.
291. //----------------INTERRUPCION EXTERNA--------------------
292. void rutina_excepcion1(void) {
293.   Flag = 1;
294.   interrupcion = 1;
295. }
296. void rutina_excepcion2(void) {
297.   Flag = 1;
298.   interrupcion = 2;
299. }
```

Programa general del Arduino Due

El programa que describimos a continuación es el que se usará de carácter general cuando la placa este completamente operativa. El que se uso para las pruebas de funcionamiento es uno similar ubicado más adelante.

```
1. //Programa escrito y diseñado por Julio Sebastián Sullca Trillo
   para el proyecto Diseño y construcción de una placa electrónica
   para el control de una bobina Tesla
2.
3. //---------variables para toma de datos-----
4. volatile int testigo;
5. volatile int Potencia;
6.
7. //------variables para el menu  -------------------------
8. char submenu;
9. char lectura;
10. int interrupcion;
11. int alerta;
12.
13. //---variables para la generadora de frecuencias-----------
14. volatile int count = 0;
15. volatile unsigned int Frec;
16. volatile int duty_cycle = 100;
17. volatile int dead_time = 350;
18. volatile unsigned int rafaga_pulsos = 100; //porcentaje en base
   100
```

```
19.
20. //-------variables para la lectura de datos--------------
21. float V;
22. float I;
23. int P;
24.
25. //---------funciones----------------------
26. void esperar_dato(void);//Espera dato por puerto serie
27. void vaciar_buffer_entrada(void);//Vacia el buffer de lectura
28. void esperar_dato2(void);//Espera dato por puerto serie del
    arduino nano
29. void vaciar_buffer_entrada2(void);//Vacia el buffer de lectura
    del arduino nano
30. void apagartimers(void);//Apaga los timers que controlan el IGBT
31. void set_frequency (unsigned int F, unsigned int duty, unsigned
    int rafaga, unsigned int d_t);
32. void toma_de_datos(void);
33. void buscar_valle(void);
34.
35. void setup() {
36.   Serial.begin(115200);//inicia comunicacion con el ordenador
37.   Serial2.begin(115200);//inicia comunicacion con el arduino
    nano
38.   pinMode(3, OUTPUT);//salida de pulsos
39.   pinMode(11, OUTPUT);//salida de pulsos
40.
41. }
42.
43. void loop() {
44.   alerta = 0;
45.   Frec = 0;
46.   apagartimers();
47.   Serial.println(F("\r\nMENU PRINCIPAL"));
48.   Serial.println(F("Introducir una B por puerto serie para
    buscar la frecuencia de minimo consumo"));
49.   Serial.println(F("Introducir una F por puerto serie para
    sintonizar una frecuencia"));
50.   Serial.println(F("Introducir una V por puerto serie para
    modificar el duty cycle y la rafaga de pulsos predeterminada"));
51.   Serial.flush();
52.   vaciar_buffer_entrada();
53.   esperar_dato();
54.   submenu = Serial.read();
55.
56.   switch (submenu) {
57.     case 'V' :
58.       Serial.println(F("\r\nIntroduce un valor del 1 al 100 para
    configurar el duty cycle"));
59.       Serial.flush();
60.       vaciar_buffer_entrada();
61.       esperar_dato();
62.       duty_cycle = Serial.parseInt();
63.       Serial.print(F("El nuevo duty es:
    ")); Serial.println(duty_cycle);
64.       Serial.flush();
65.
66.       Serial.println(F("\r\nIntroduce un valor del 1 al 100 para
    configurar la rafaga de pulsos en base 100"));
67.       Serial.flush();
68.       vaciar_buffer_entrada();
69.       esperar_dato();
```

```
70.       rafaga_pulsos = Serial.parseInt();
71.       Serial.print(F("La nueva rafaga es:
   ")); Serial.println(rafaga_pulsos);
72.       Serial.flush();
73.
74.       break;
75.
76.     case 'B':
77.       Serial.println(F("\r\nIntroduce una frecuencia de partida
   sin puntos ni comas, un 0 te devuelve al menu principal"));
78.       Serial.flush();
79.       vaciar_buffer_entrada();
80.       esperar_dato();
81.       Frec = Serial.parseInt();
82.       Serial.print(F("Frecuencia escogida:
   ")); Serial.println(Frec); //testpoint
83.       if (Frec != 0) {
84.         buscar_valle();//Funcion que busca la frecuencia de
   minimo consumo
85.         Serial.print(F("La frecuencia de valle de la bobina de
   Tesla es:")); Serial.println(Frec);
86.         Serial.println(F("volvemos al menu principal"));
87.         Serial.flush();
88.       }
89.       else {
90.         Serial.println(F("volvemos al menu principal"));
91.         Serial.flush();
92.       }
93.       break;
94.
95.     case 'F':
96.       Serial.println(F("\r\nEn este menu sintonizamos una
   frecuencia, veremos por pantalla los datos de la placa"));
97.       Serial.println(F("Solo se debe introducir la frecuencia
   deseada por puerto serie sin puntos ni comas"));
98.       Serial.println(F("Un 0 nos devuelve al menu principal,las
   protecciones nos devuelven al menu principal"));
99.       Serial.println(F("El inversor permanecera apagado hasta
   que introduzcamos la primera frecuencia"));
100.      Serial.flush();
101.      vaciar_buffer_entrada();
102.      vaciar_buffer_entrada2();
103.      while (1) {
104.        if (Serial2.available()) {
105.
106.          lectura = Serial2.read();
107.          switch (lectura) {
108.            case 'N':
109.              delayMicroseconds(500);//esperemos que la
   comunicacion se complete
110.              interrupcion = Serial2.parseInt();
111.              Serial.print(F("Ha ocurrido el fallo:
   ")); Serial.println(interrupcion);
112.              alerta = 1;
113.              break;
114.
115.            case 'Y':
116.              delayMicroseconds(500);//esperamos que la
   comunicacion se complete
117.              I = Serial2.parseFloat();
118.              V = Serial2.parseFloat();
```

```
119.               P = Serial2.parseInt();
120.               Serial2.read();//Aqui lee la F de final de mensaje
121.               vaciar_buffer_entrada2();//Al acabar de leer un
   mensaje deberiamos vaciar el buffer por si acaso
122.               Serial.print("Frecuencia:
   "); Serial.println(Frec);
123.               Serial.print("I: "); Serial.println(I);
124.               Serial.print("V: "); Serial.println(V);
125.               Serial.print("P: "); Serial.println(P);
126.               Serial.flush();
127.               if (Serial.available()) {
128.                 Frec = Serial.parseInt();
129.                 vaciar_buffer_entrada();
130.                 Serial.print(F("Frecuencia escogida:
   ")); Serial.println(Frec); //testpoint
131.                 if (Frec == 0) {
132.                   alerta = 1;
133.                 }
134.                 else {
135.                   set_frequency (Frec, duty_cycle,
   rafaga_pulsos, dead_time);
136.                 }
137.
138.               }
139.               break;
140.
141.             default:
142.               //No hacemos nada sigue leyendo
143.               break;
144.           }
145.         }
146.         if (alerta == 1) { //Volvemos al menu principal si salto
   una proteccion o si introdujimos un 0
147.           break;
148.         }
149.       }
150.       break;
151.
152.     default:
153.       Serial.println(F("\r\nNo te he entendido"));
154.       Serial.flush();
155.       break;
156.
157.   }
158. }
159. //FUNCIONES BASICAS PARA LA COMUNICACION SERIE
160. //
161. //
162. void esperar_dato(void) {//espera un dato por el puerto serie
163.   while (Serial.available() == 0) {}
164.   delay(70);
165. }
166. void vaciar_buffer_entrada(void) {//Vacia el buffer de lectura
167.   while (Serial.available() != 0) {
168.     Serial.read();
169.   }
170. }
171.
172. void esperar_dato2(void) {//espera un dato por el puerto serie
   del arduino nano
173.   while (Serial2.available() == 0) {}
```

```
174.   delay(1);
175. }
176. void vaciar_buffer_entrada2(void) {//Vacia el buffer de lectura
   del arduino nano
177.   while (Serial2.available() != 0) {
178.     Serial2.read();
179.   }
180. }
181.
182. void apagartimers(void) {
183.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID34;                    // TC7
   power OFF - Timer Counter 2 channel 1 IS TC7 - See page 38 pin 3
184.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID35;                    // TC8
   power OFF - Timer Counter 2 channel 2 IS TC8 - See page 38 pin 11
185. }
186.
187.
188. //FUNCION PARA LA TOMA DE DATOS
189.
190. void toma_de_datos(void) { // ESTA FUNCION SOLO TOMA EL DATO DEL
   ESTADO Y DE LA POTENCIA
191.   char M;
192.   vaciar_buffer_entrada2();//Limpia el puerto del arduino nano
193.   while ((Serial2.read()) != 'F') {} //Leer hasta encontrar un
   principio de mensaje
194.   Serial2.read();//Esperar al asegunda lista de valores
195.   while ((Serial2.read()) != 'F') {} //Leer hasta encontrar un
   principio de mensaje
196.   esperar_dato2();
197.   M = Serial2.read();//lee el inicio del mensaje
198.
199.   switch (M) {
200.
201.     case 'Y':
202.
203.       testigo = 0; //No ha habido errores
204.       Serial2.parseFloat();//aqui lee la corriente
205.       Serial2.parseFloat();//aqui lee el valor de la tension
206.       Potencia = Serial2.parseInt(); //aqui lee la potencia
207.       vaciar_buffer_entrada2();//vaciamos buffer
208.
209.       break;
210.
211.     case 'N':
212.
213.       testigo = Serial2.parseInt(); //almacenamos el error
214.       Potencia = 20000; //esta apagado la potencia es 0 pero le
   ponemos un valor maximo e inalcanzable
215.       vaciar_buffer_entrada2();//vaciamos buffer
216.
217.       break;
218.
219.     default:
220.       Serial.println(F("No he entendido nada, reseteo el
   sketch"));//IMPORTANTE SI ESTO OCURRE HAY PROBLEMAS DE LECTURA
221.       Serial.flush();
222.
223.       RSTC->RSTC_CR = 0xA5000005; // Reset processor and
   internal peripherals
224.       break;
225.   }
```

```
226.
227. }
228. //FUNCION PARA BUSCAR VALLE DE POTENCIA
229.
230. void buscar_valle(void) {
231.   //---VARIABLES PARA FUNCION BUSCAR VALLE-----
232.   volatile int E0;//frecuencia inicial
233.   volatile int E1;//frecuencia -paso
234.   volatile int E2;//frecuencia +paso
235.   volatile int P0;//frecuencia inicial
236.   volatile int P1;//frecuencia -paso
237.   volatile int P2;//frecuencia +paso
238.   int paso = 1600; //Paso de la frecuencia que utilizo
239.   int iteracion = 0; //veces que le he recorrido el bucle en
   cada iteracion recorremos 3 frecuencias como mucho medio segundo
240.   int menor;//La frecuencia con menor potencia
241.   //VARIABLES PARA 3 LECTURAS DE POTENCIA
242.
243.
244.   while ((paso != 50) && (iteracion <= 300)) { //como maximo
   habra 300 iteracines aproximamos cada iteracion de forma mayorada
   a medio segundo
245.     iteracion++;//aumentamos en 1 la iteracion
246.     set_frequency (Frec, duty_cycle, rafaga_pulsos,
   dead_time); //probamos con la frecuencia de partida para esta
   iteracion
247.     toma_de_datos();//sacamos la informacion del nano
248.     E0 = testigo;
249.     P0 = Potencia;
250.     //-----------sacamos por pantalla los datos de esta
   iteracion---------------
251.     Serial.print("\r\n"); Serial.print(iteracion); Serial.printl
   n("° iteracion ");
252.     Serial.print("Frecuencia:"); Serial.println(Frec);
253.     Serial.print("Paso:"); Serial.println(paso);
254.     Serial.print("Estado:"); Serial.println(E0);
255.     Serial.print("Potencia: "); Serial.println(P0);
256.     //-------------------------------------------------------
257.
258.     set_frequency ((Frec - paso), duty_cycle, rafaga_pulsos,
   dead_time); //probamos un paso por debajo de la frecuencia
259.     toma_de_datos();//sacamos la informacion del nano
260.     E1 = testigo;
261.     P1 = Potencia;
262.     Serial.print("Estado-PASO:"); Serial.println(E1);
263.     Serial.print("Potencia-PASO: "); Serial.println(P1);
264.     set_frequency ((Frec + paso), duty_cycle, rafaga_pulsos,
   dead_time); //probamos un paso por arriba de la frecuencia
265.     toma_de_datos();//sacamos la informacion del nano
266.     E2 = testigo;
267.     P2 = Potencia;
268.     Serial.print("Estado+PASO:"); Serial.println(E2);
269.     Serial.print("Potencia+PASO: "); Serial.println(P2);
270.
271.     //---------respuesta por sobretemperatura-----------------
272.     if ((E0 == 3) || (E1 == 3) || (E2 == 3)) { //respuesta si en
   algun momento la temperatura ha subido en exceso, despues de la
   primera iteracion es improbable que ocurra
273.       Serial.println(F("SOBRETEMPERATURA!!, reseteo el
   sketch"));
274.       Serial.flush();
```

```
275.       RSTC->RSTC_CR = 0xA5000005; // Reset processor and
   internal peripherals
276.     }
277.     //---------respuesta por sobrecorriente--------------------
   --------
278.     if ((P0 == 20000) && (P1 == 20000) && (P2 == 20000)) { //res
   puesta si en algun momento la corriente ha subido en exceso,
   despues de la primera iteracion es improbable que ocurra
279.       Serial.println(F("LA FRECUENCIA ESCOGIDA ESTA EN UN PUNTO
   DE OPERACION PELIGROSO, reseteo el sketch"));
280.       Serial.flush();
281.       RSTC->RSTC_CR = 0xA5000005; // Reset processor and
   internal peripherals
282.     }
283.
284.     //Buscamos el valor minimo de potencia con que frecuencia se
   ha alcanzado
285.     if (P0 > P1) {
286.       if (P1 > P2) {
287.         menor = 2;
288.       }
289.       else {
290.         menor = 1;
291.       }
292.     }
293.     else {
294.       if (P0 > P2) {
295.         menor = 2;
296.       }
297.       else {
298.         menor = 0;
299.       }
300.     }
301.
302.     //Una vez tenemos la frecuencia de potencia menor.........
303.     switch (menor) {
304.       case 0://Si es la misma hay que reducir el paso de la
   siguiente iteracion
305.         paso = paso / 2;
306.         break;
307.       case 1://La frecuencia es la actual menos el paso actual
308.         Frec = Frec - paso;
309.         break;
310.       case 2://La frecuencia es la actual mas el paso actual
311.         Frec = Frec + paso;
312.         break;
313.     }
314.
315.   }
316.
317.   apagartimers();
318. }
319.
320. //FUNCION PARA ESTABLECER UNA FRECUENCIA
321.
322. void set_frequency (unsigned int F, unsigned int duty, unsigned
   int rafaga, unsigned int d_t) { //d_t en nanosegundos.//duty y
   rafaga en base 100
323.   unsigned int N;
324.   float D;
325.   int A = 0;
```

```
326.   int D_corregido;
327.   //int T_duty=0;//parametro para la modulacion del pulso 4%
   ESTA BIEN
328.   int operacion;//parametro auxiliar
329.   count = 0;
330.   N = 40000000 / (F * 0.952381 * 0.999);
331.   D = d_t / 23.78571548;
332.   operacion = (N / 2.0) - ((duty / 100.0) * (N / 2.0));
333.   if (operacion > D) {
334.     D = operacion;
335.   }
336.   D_corregido = D - 10; // minimo 261.6428 ns en torno 300
337.   NVIC_DisableIRQ(TC8_IRQn);//desactivamos interrupciones
338.
339.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID34;                      // TC7
   power OFF - Timer Counter 2 channel 1 IS TC7 - See page 38 pin 3
340.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID35;                      // TC8
   power OFF - Timer Counter 2 channel 2 IS TC8 - See page 38 pin 11
341.   TC2->TC_CHANNEL[1].TC_CCR = (0 << 2) | (1 << 1) | (0 << 0); //
   Software trigger TC7--0 counter and disable
342.   TC2->TC_CHANNEL[2].TC_CCR = (0 << 2) | (1 << 1) | (0 << 0); //
   Software trigger TC8--0 counter and disable
343.   //Set up del timer que nos dara los tiempos
344.   PMC->PMC_PCER1 |= PMC_PCER1_PID34;                      // TC7
   power ON - Timer Counter 2 channel 1 IS TC7 - See page 38
345.   PMC->PMC_PCER1 |= PMC_PCER1_PID35;                      // TC8
   power ON - Timer Counter 2 channel 2 IS TC8 - See page 38
346.
347.   PIOC->PIO_PDR |= PIO_PDR_P28;                           // The
   pin is no more driven by GPIO TIOA7
348.   PIOD->PIO_PDR |= PIO_PDR_P7;                           // The
   pin is no more driven by GPIO TIOA8
349.
350.   PIOC->PIO_ABSR |= PIO_PC28B_TIOA7;                      //
   Periperal type B  - See page 859
351.   PIOD->PIO_ABSR |= PIO_PD7B_TIOA8;                      //
   Periperal type B  - See page 859
352.
353.   //TIOA7=pin 3
354.   TC2->TC_CHANNEL[1].TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1  //
   MCK/2, clk on rising edge
355.                               | TC_CMR_WAVE               //
   Waveform mode
356.                               | TC_CMR_WAVSEL_UP_RC        // UP
   mode with automatic trigger on RC Compare
357.                               | TC_CMR_ACPA_SET           // Set
   TIOA7 on RA compare match  -- See page 883
358.                               | TC_CMR_ACPC_CLEAR;          //
   Clear TIOA7 on RC compare match
359.
360.   //TIOA8=pin 11
361.   TC2->TC_CHANNEL[2].TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1  //
   MCK/2, clk on rising edge
362.                               | TC_CMR_WAVE               //
   Waveform mode
363.                               | TC_CMR_WAVSEL_UP_RC        // UP
   mode with automatic trigger on RC Compare
364.                               | TC_CMR_ACPA_CLEAR          //
   Clear TIOA8 on RA compare match  -- See page 883
365.                               | TC_CMR_ACPC_SET;           //
   Set TIOA8 on RC compare match
```

```
366.
367.   TC2->TC_CHANNEL[1].TC_RC = N;  //<********************  Frequency = (Mck/2)/TC_RC  Hz
368.   TC2->TC_CHANNEL[1].TC_RA = ((N / 2) + D); //<********************   T_OFF TIOA7 [25 ns/unidad]
369.
370.   TC2->TC_CHANNEL[2].TC_RC = N;  //<********************  Frequency = (Mck/2)/TC_RC  Hz
371.   TC2->TC_CHANNEL[2].TC_RA = ((N / 2) - D); //<********************  T_ON TIOA8 [25 ns/unidad]
372.
373.
374.
375.   if (rafaga < 100) {
376.     TC2->TC_CHANNEL[2].TC_IER = TC_IER_CPCS;                  // Interrupt on RC compare match
377.
378.     NVIC_EnableIRQ(TC8_IRQn);
379.   }
380.
381.   TC2->TC_CHANNEL[1].TC_CCR = TC_CCR_SWTRG | TC_CCR_CLKEN; // Software trigger TC7 counter and enable
382.
383.
384.   while ( A < (D_corregido)) {
385.     A = TC2->TC_CHANNEL[1].TC_CV;
386.     //PERMANECEMOS AQUI MIENTRAS ESPERAMOS EL DEADTIME
387.   }
388.
389.   TC2->TC_CHANNEL[2].TC_CCR = TC_CCR_SWTRG | TC_CCR_CLKEN; // Software trigger TC8 counter and enable
390. }
391.
392.
393. //-----------------INTERRUPCION-----------------------
394.
395. void TC8_Handler() {
396.
397.   count++;
398.
399.   if (count >= rafaga_pulsos) {
400.     PIOD->PIO_PER |= PIO_PER_P7;                              // The pin is  driven by GPIO TIOA8
401.     PIOD->PIO_CODR = (1 << 7) ;
402.     PIOC->PIO_PER |= PIO_PER_P28;                             // The pin is  driven by GPIO TIOA7
403.     PIOC->PIO_CODR = (1 << 28) ;
404.   }
405.   if (count >= 100) { //en base 100
406.     count = 0;
407.     PIOC->PIO_PDR |= PIO_PDR_P28;                             // The pin is NOT driven by GPIO TIOA7
408.     PIOD->PIO_PDR |= PIO_PDR_P7;   // The pin is NOT  driven by GPIO TIOA8
409.   }
410.   //}
411.   TC2->TC_CHANNEL[2].TC_SR;
```

```
412. }
```

Programa para caso particular del Arduino Due

Como se describe en la memoria. La placa no pudo estar completamente operativa durante las pruebas de funcionamiento. Faltó por implementar el módulo del Arduino Nano. El programa general del Arduino Due está diseñado por seguridad para funcionar solo si hay lecturas del Arduino Nano. Se modifico el programa del Arduino Due para que produjese una respuesta sin realimentación de control. Este es un bucle abierto donde el control se ejecuta manualmente actuando directamente sobre el Timer del Arduino Due. Se pueden modificar las consignas de este a través del puerto serie del ordenador. Los Timers son controlados por la función "set_frequency" El programa es muy similar al anterior con una interfaz muy intuitiva.

```
1. //#define PIN_CORTE 50
2. volatile int count = 0;
3. volatile unsigned int rafaga_pulsos=100;//porcentaje en base 100
4. void set_frequency (unsigned int F,unsigned int duty,unsigned int
    rafaga,unsigned int d_t);
5. void esperar_dato(void);
6. void vaciar_buffer_entrada(void);
7. void apagartimers(void);
8.
9.
10. unsigned int frecuencia;
11. unsigned int duty_cycle;
12. unsigned int dead_time;
13.
14. void setup() {
15. ///pinMode(PIN_CORTE, OUTPUT);
16. pinMode(3, OUTPUT);
17. pinMode(11, OUTPUT);
18. Serial.begin(115200);
19.
20. }
21. void loop() {
22. apagartimers();
23. Serial.println(F(""));
24. Serial.println(F("INTRODUCE DATOS:"));
25. Serial.println(F("LEO FRECUENCIA EN HZ SIN COMAS NI PUNTOS"));
26. Serial.flush();
27. vaciar_buffer_entrada();
28. esperar_dato();
29. frecuencia=Serial.parseInt();
30. Serial.println(F("LEO DUTY CYCLE EN PORCENTAJE DEL 1 AL 100 SOLO
   ENTEROS"));
31. Serial.flush();
32. vaciar_buffer_entrada();
33. esperar_dato();
34. duty_cycle=Serial.parseInt();
35. Serial.println(F("LEO RAFAGA, PULSOS QUE SE DEJAN PASAR POR CADA
   RAFAGA DE 100, VALORES DEL 1 AL 100 SOLO"));
36. Serial.flush();
37. vaciar_buffer_entrada();
38. esperar_dato();
39. rafaga_pulsos=Serial.parseInt();
40. Serial.println(F("LEO DEADTIME EN NS MINIMO 275"));
41. Serial.flush();
42. vaciar_buffer_entrada();
43. esperar_dato();
```

```
44. dead_time=Serial.parseInt();
45. Serial.println(F("LOS DATOS INTRODUCIDOS SON:"));
46. Serial.print(F("FRECUENCIA: "));Serial.println(frecuencia);
47. Serial.print(F("DUTY: "));Serial.println(duty_cycle);
48. Serial.print(F("RAFAGA:"));Serial.println(rafaga_pulsos);
49. Serial.print(F("TIEMPO MUERTO:"));Serial.println(dead_time);
50. set_frequency (frecuencia,duty_cycle,rafaga_pulsos,dead_time);
51. while(1){
52.  esperar_dato();
53.  if (Serial.read()=='B'){
54.   break;
55.  }
56. }
57. }
 58.
59. void set_frequency (unsigned int F,unsigned int duty,unsigned in
   t rafaga,unsigned int d_t) { //d_t en nanosegundos.//duty y
   rafaga en base 100
60.   unsigned int N;
61.   float D;
62.   int A = 0;
63.   int D_corregido;
64.   //int T_duty=0;//parametro para la modulacion del pulso 4%
   ESTA BIEN
65.   int operacion;//parametro auxiliar
66.   count=0;
67.   N = 40000000/ (F*0.952381*0.999);
68.   D = d_t/23.78571548;
69.   operacion= (N/2.0)-((duty/100.0)*(N/2.0));
70.   if (operacion>D){
71.   D=operacion;
72.   }
73.   D_corregido = D-10;// minimo 261.6428 ns ANTES ERA 11
74.   NVIC_DisableIRQ(TC8_IRQn);//desactivamos interrupciones
75.
76.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID34;                      // TC7
   power OFF - Timer Counter 2 channel 1 IS TC7 - See page 38 pin 3
77.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID35;                      // TC8
   power OFF - Timer Counter 2 channel 2 IS TC8 - See page 38 pin 11
78.   TC2->TC_CHANNEL[1].TC_CCR = (0<<2)|(1<<1)|(0<<0); // Software
   trigger TC7--0 counter and disable
79.   TC2->TC_CHANNEL[2].TC_CCR = (0<<2)|(1<<1)|(0<<0); // Software
   trigger TC8--0 counter and disable
80.   //Set up del timer que nos dara los tiempos
81.   PMC->PMC_PCER1 |= PMC_PCER1_PID34;                      // TC7
   power ON - Timer Counter 2 channel 1 IS TC7 - See page 38
82.   PMC->PMC_PCER1 |= PMC_PCER1_PID35;                      // TC8
   power ON - Timer Counter 2 channel 2 IS TC8 - See page 38
83.
84.   PIOC->PIO_PDR |= PIO_PDR_P28;                           // The
   pin is no more driven by GPIO TIOA7
85.   PIOD->PIO_PDR |= PIO_PDR_P7;                           // The
   pin is no more driven by GPIO TIOA8
86.
87.   PIOC->PIO_ABSR |= PIO_PC28B_TIOA7;                      //
   Periperal type B  - See page 859
88.   PIOD->PIO_ABSR |= PIO_PD7B_TIOA8;                      //
   Periperal type B  - See page 859
89.
90.   //TIOA7=pin 3
```

```
91.    TC2->TC_CHANNEL[1].TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1  //
   MCK/2, clk on rising edge
92.                                | TC_CMR_WAVE                //
   Waveform mode
93.                                | TC_CMR_WAVSEL_UP_RC        // UP
   mode with automatic trigger on RC Compare
94.                                | TC_CMR_ACPA_SET           // Set
   TIOA7 on RA compare match  -- See page 883
95.                                | TC_CMR_ACPC_CLEAR;          //
   Clear TIOA7 on RC compare match
96.
97.     //TIOA8=pin 11
98.    TC2->TC_CHANNEL[2].TC_CMR = TC_CMR_TCCLKS_TIMER_CLOCK1  //
   MCK/2, clk on rising edge
99.                                | TC_CMR_WAVE                //
   Waveform mode
100.                               | TC_CMR_WAVSEL_UP_RC        // UP
   mode with automatic trigger on RC Compare
101.                               | TC_CMR_ACPA_CLEAR          //
   Clear TIOA8 on RA compare match  -- See page 883
102.                               | TC_CMR_ACPC_SET;           //
   Set TIOA8 on RC compare match
103.
104.   TC2-
   >TC_CHANNEL[1].TC_RC = N;  //<*********************  Frequency =
   (Mck/2)/TC_RC  Hz
105.   TC2-
   >TC_CHANNEL[1].TC_RA = ((N/2)+D);  //<********************   T_OFF
   TIOA7 [25 ns/unidad]
106.
107.   TC2-
   >TC_CHANNEL[2].TC_RC = N;  //<*********************  Frequency =
   (Mck/2)/TC_RC  Hz
108.   TC2->TC_CHANNEL[2].TC_RA = ((N/2)-
   D);  //<*********************  T_ON TIOA8 [25 ns/unidad]
109.
110.
111.
112.   if(rafaga<100){
113.   TC2->TC_CHANNEL[2].TC_IER = TC_IER_CPCS;                 //
   Interrupt on RC compare match
114.
115.   NVIC_EnableIRQ(TC8_IRQn);
116.   }
117.
118.   TC2->TC_CHANNEL[1].TC_CCR = TC_CCR_SWTRG | TC_CCR_CLKEN; //
   Software trigger TC7 counter and enable
119.
120.
121.   while ( A < (D_corregido)){
122.    A= TC2->TC_CHANNEL[1].TC_CV;
123.   //PERMANECEMOS AQUI MIENTRAS ESPERAMOS EL DEADTIME
124.   }
125.
126.   TC2->TC_CHANNEL[2].TC_CCR = TC_CCR_SWTRG | TC_CCR_CLKEN; //
   Software trigger TC8 counter and enable
127. }
128.
129.
130.
131.
```

```
132. //-------FUNCIONES BASICAS PARA LA COMUNICACION SERIE------
133.
134. void esperar_dato(void) {//espera un dato por el puerto serie
135.   while (Serial.available() == 0) {}
136.   delay(70);
137. }
138. void vaciar_buffer_entrada(void) {//Vacia el buffer de lectura
139.   while (Serial.available() != 0) {
140.     Serial.read();
141.   }
142. }
143.
144.
145. void apagartimers(void){
146.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID34;                      // TC7
   power OFF - Timer Counter 2 channel 1 IS TC7 - See page 38 pin 3
147.   PMC->PMC_PCDR1 |= PMC_PCDR1_PID35;                      // TC8
   power OFF - Timer Counter 2 channel 2 IS TC8 - See page 38 pin 11
148. }
149.
150.
151.
152.
153. //-----------------INTERRUPCION-----------------------
154.
155. void TC8_Handler() {
156.
157.   count++;
158.
159.   if (count>=rafaga_pulsos){
160.   PIOD->PIO_PER |= PIO_PER_P7;                          // The
   pin is  driven by GPIO TIOA8
161.   PIOD->PIO_CODR=(1<<7) ;
162.   PIOC->PIO_PER |= PIO_PER_P28;                          // The
   pin is  driven by GPIO TIOA7
163.   PIOC->PIO_CODR=(1<<28) ;
164.   }
165.   if(count>=100){//en base 100
166.   count=0;
167.   PIOC->PIO_PDR |= PIO_PDR_P28;                          // The
   pin is NOT driven by GPIO TIOA7
168.   PIOD->PIO_PDR |= PIO_PDR_P7;   // The pin is NOT  driven by
   GPIO TIOA8
169.   }
170.   //}
171.    TC2->TC_CHANNEL[2].TC_SR;
172. }
```

Anexo III: Hoja de datos de la placa

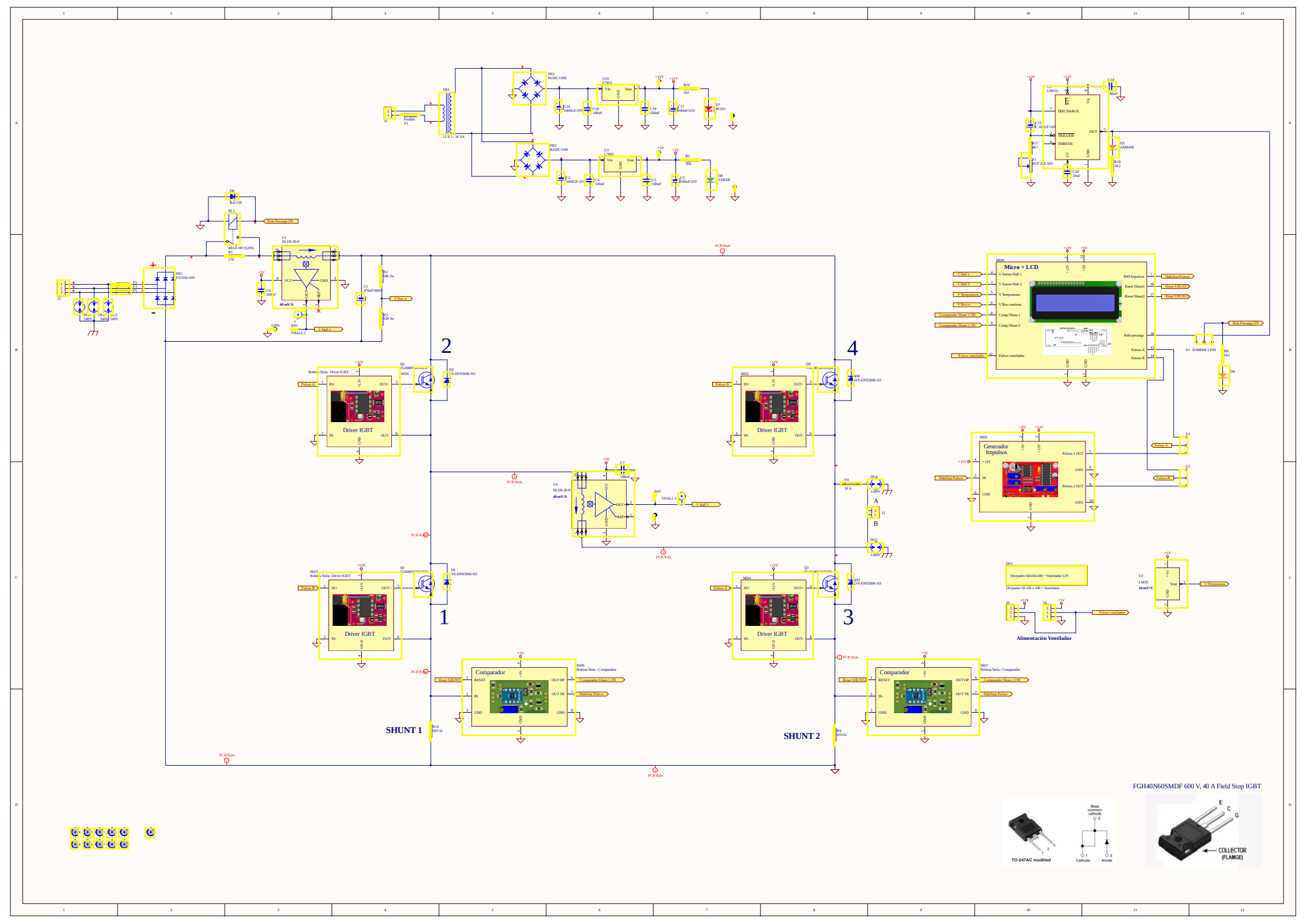
En la siguiente tabla se recogen una serie de parámetros que caracterizan el control de la placa y la electrónica de potencia.

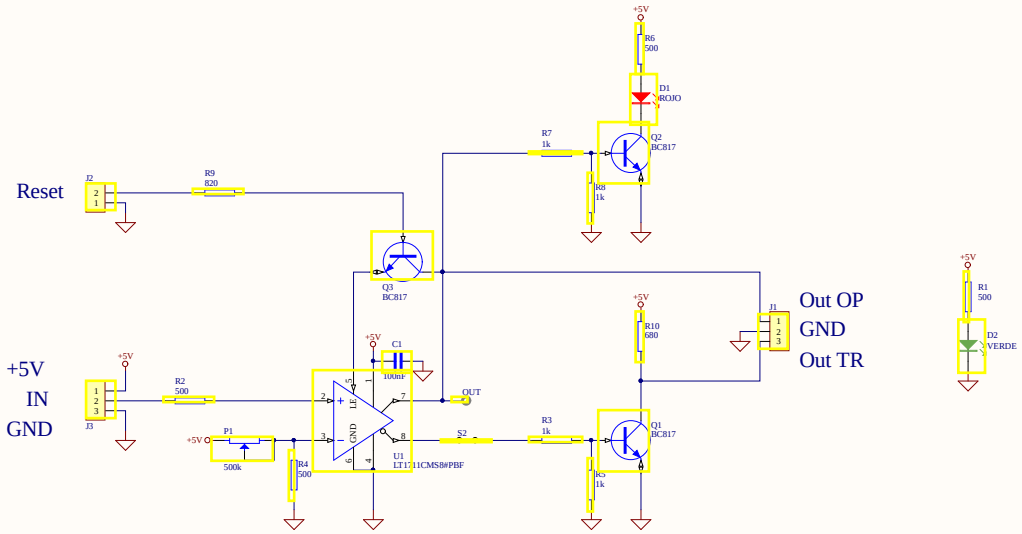
Valores característicos			
Corriente instantánea máxima	40 [A]	Sensibilidad en lectura de corriente	0,04[V_{ADC}/A]
Corriente eficaz máxima	10 [A]	Sensibilidad en lectura de tensión	0,0119[V_{ADC}/ V_{BUS}]
Tensión máxima en el bus de continua	400[V]	Sensibilidad en medida de temperatura	0,1[V_{ADC}/ºC]
Tensión máxima eficaz de entrada	270[V]	Periodo ciclo de control del Arduino nano	100[ms]
Potencia monofásica (rizado del 15%, 230 V)	700 [W]	Anchura mínima para el pulso de ON del transistor	833 [ns]
Potencia trifásica (rizado del 15%, 230 V de línea)	2100 [W]	Dead time predeterminado	350 [ms]
Potencia térmica disipada (200 kHz, alimentación monofásica a 230V y 10A eficaces)	61,8 [W]	Rango de frecuencias óptimo/ dead time mínimo/tolerancia	Hasta 200 [kHz], 300[ns],±10[ns]
Velocidad de sensado	10 [KS/s]	Temperatura máxima en sensor	98[ºC]

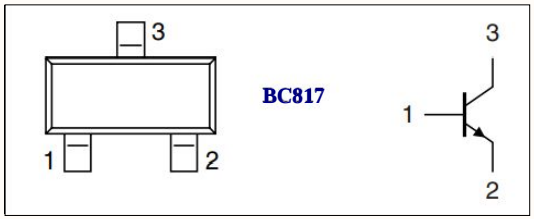
Anexo IV: Planos

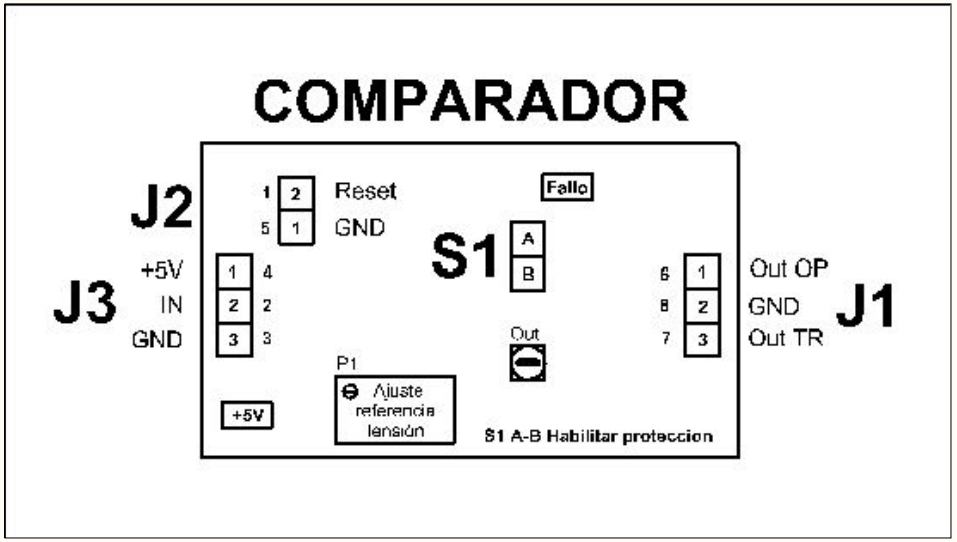
En este último anexo veremos los esquemas eléctricos de la placa de potencia y de los módulos diseñados. Los planos aparecen en el siguiente orden:

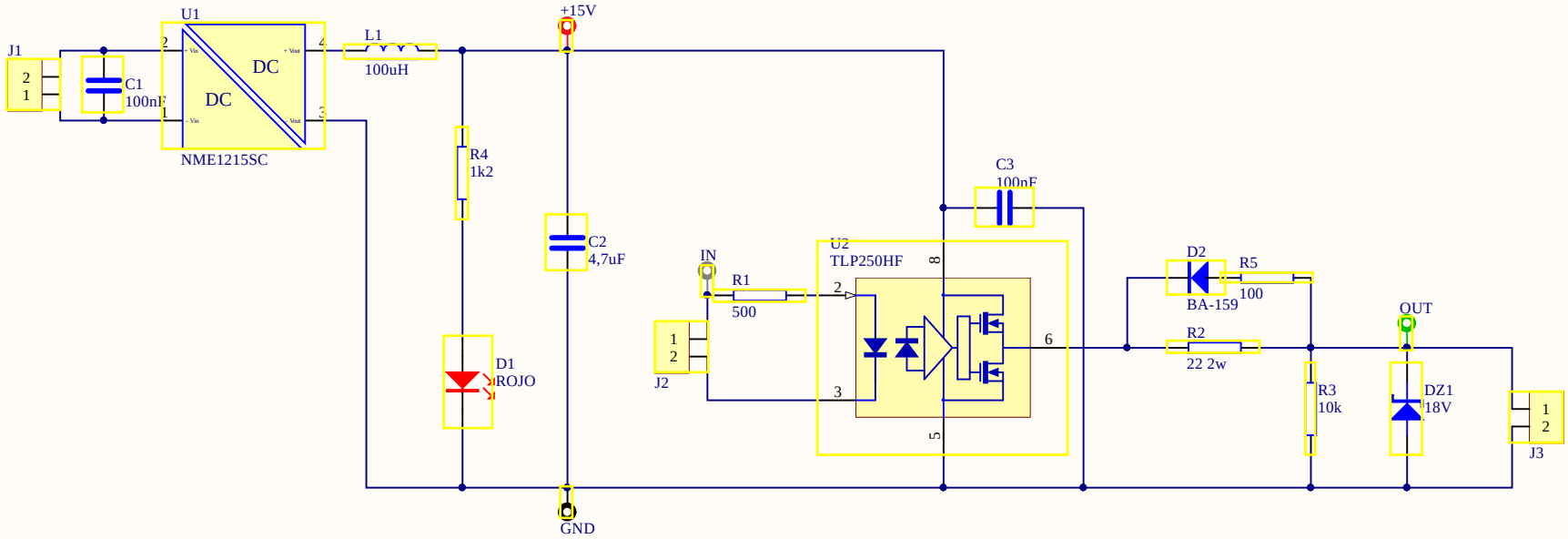
- Esquema de la placa de potencia
- Esquema del módulo de sobrecorrientes
- Esquema del módulo de disparos del IGBT
- Esquema del módulo generador de pulsos analógico
- Esquema del módulo de fibra óptica
- Esquema del módulo del Arduino Nano
- Esquema del módulo del Arduino Due

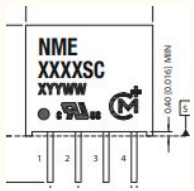

Micro + LCD
Driver IGBT
Comparador
Generador Impulsos
SHUNT 1
SHUNT 2
Alimentación Ventilador
FGH40N60SMDF 600 V, 40 A Field Stop IGBT
TO-247AC modified
Base common cathode
Cathode
Anode
COLLECTOR (FLANGE)


Reset
+5V
IN
GND
Out OP
GND
Out TR
R9
820
R7
1k
R6
500
D1
ROJO
Q2
BC817
R8
1k
Q3
BC817
C1
100nF
R10
680
R2
500
R3
1k
R4
500
P1
500k
S2
U1
LT1711CMS8#PBF
Q1
BC817
R5
1k
J1
J2
J3
R1
500
D2
VERDE

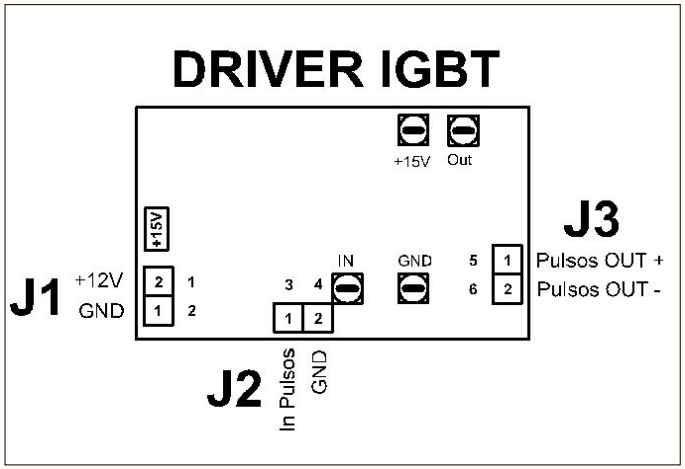

3
1
2
BC817
3
1
2


COMPARADOR
J2
Reset
GND
J3
+5V
IN
GND
S1
A
B
Fallo
Out
P1
Ajuste referencia tensión
+5V
S1 A-B Habilitar proteccion
Out OP
GND
Out TR
J1



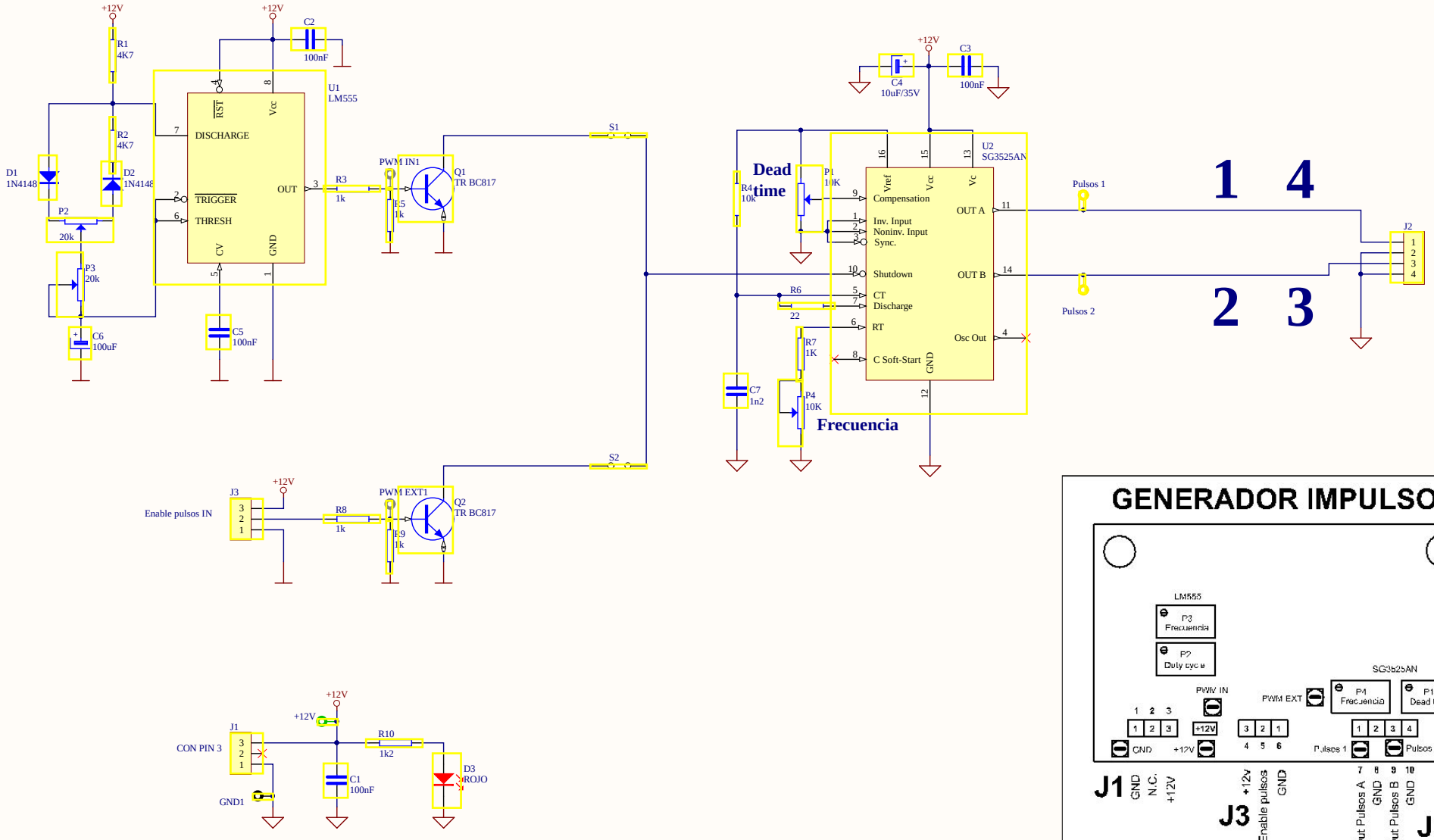


Pin	Function
1	-VIN
2	+VIN
3	-VOUT
4	+VOUT



f=0,69/((2*P1+P2+4,7k)*C1)

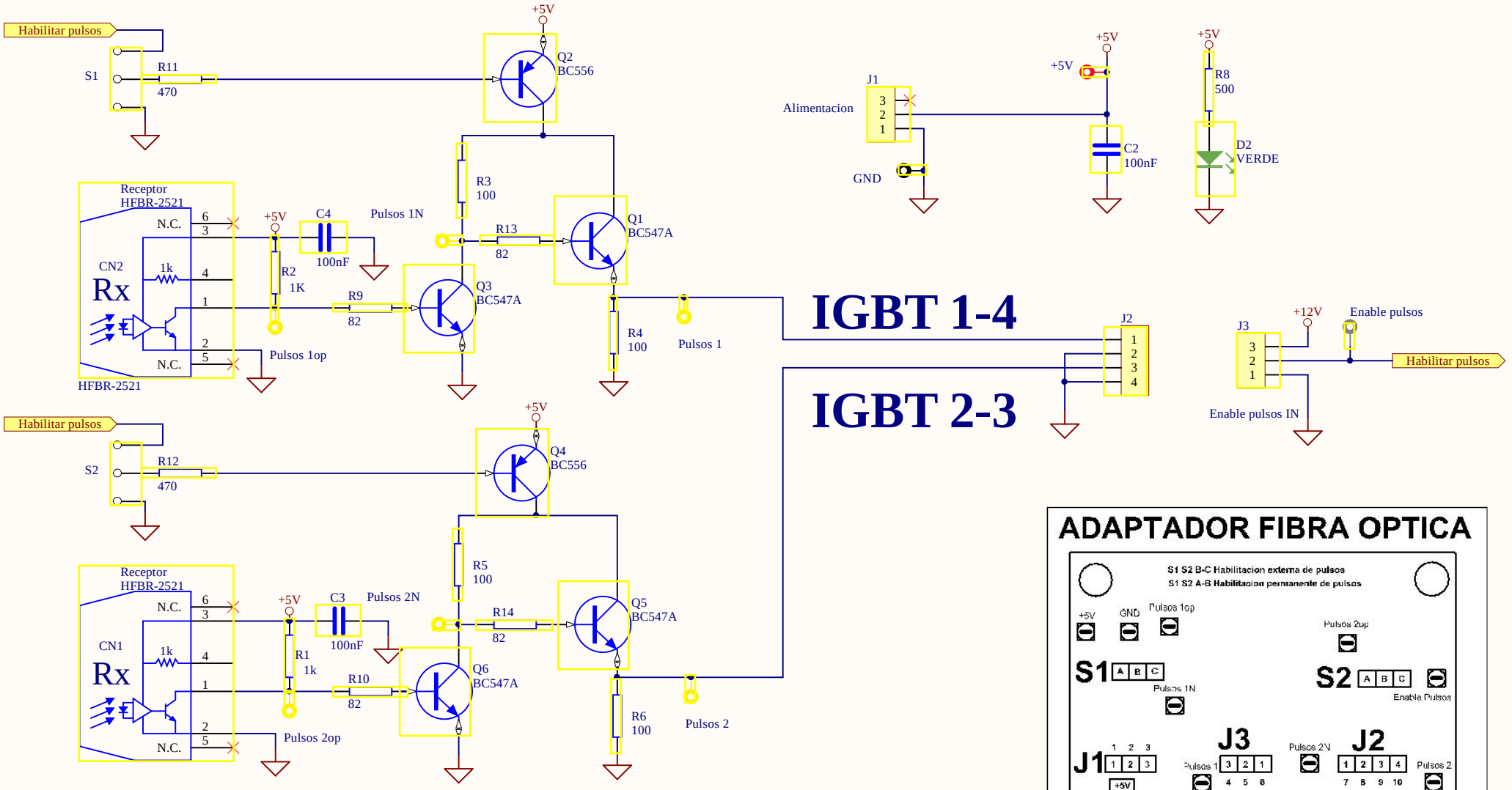
n=1+P2/P1

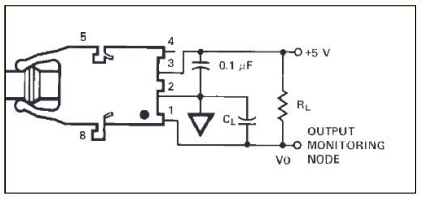


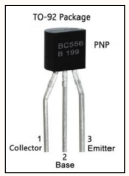
Title		
Size A3	Number	Revision
Date: 26/11/2021	Sheet of	
File: G:\Mi unidad\..\Generador impulsos V2.SchDoc	Drawn By:	

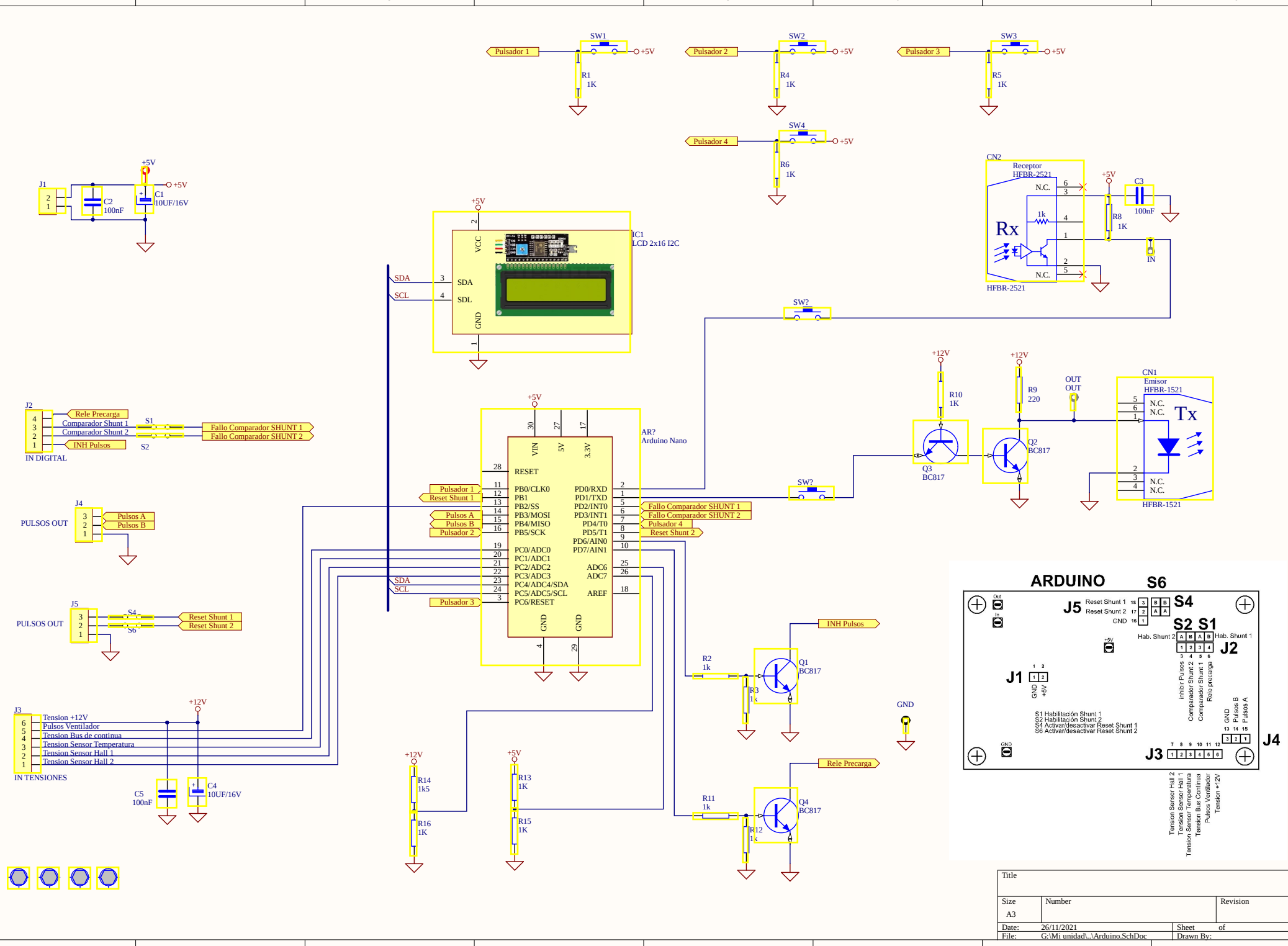
Interface Fibra optica V2

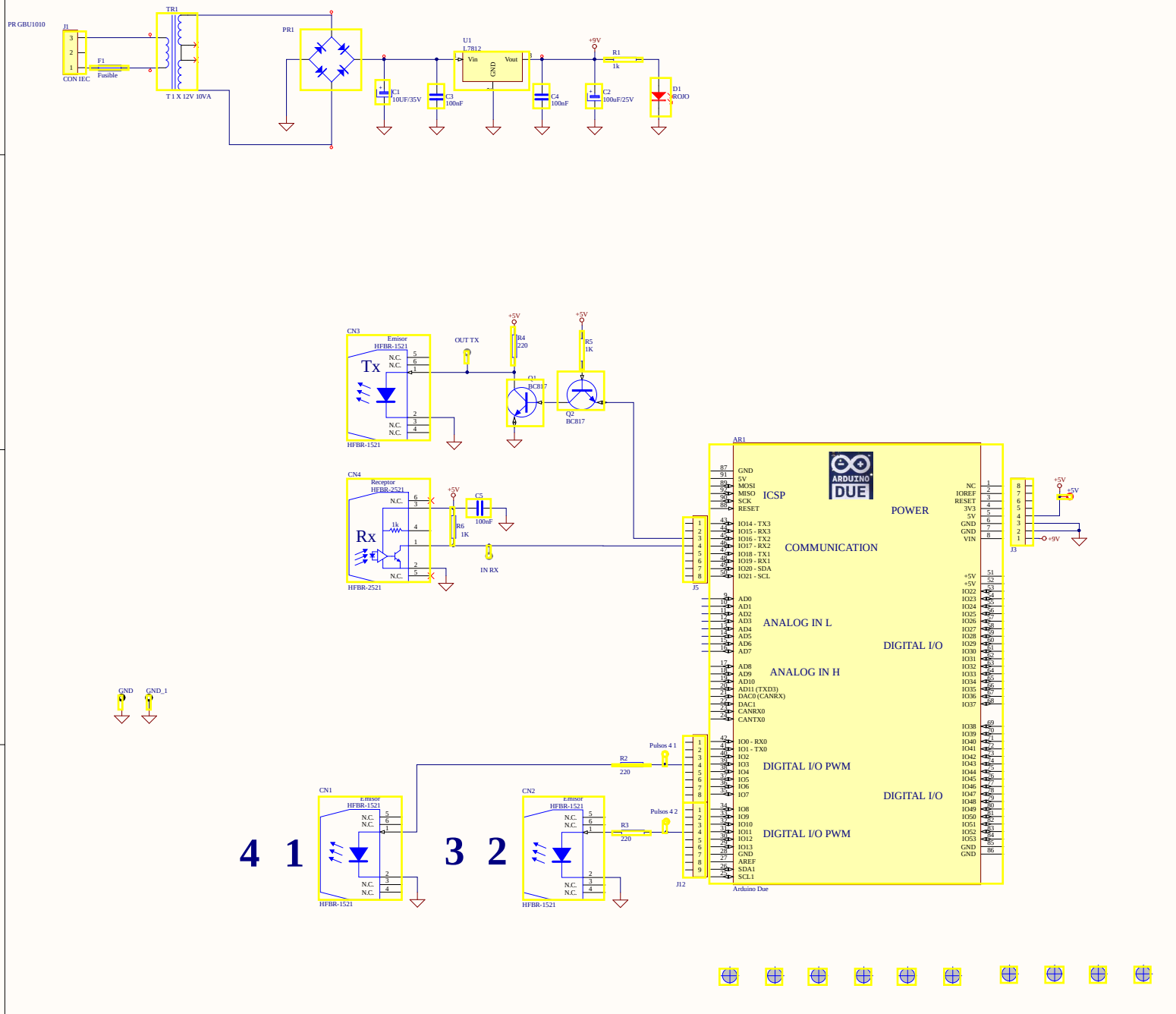
S1 S2 Permite habilitar permanentemente los pulsos prescindiendo de las protecciones externas al modulo








SW1
Pulsador 1
+5V
R1
1K
SW2
Pulsador 2
R4
1K
SW3
Pulsador 3
R5
1K
SW4
Pulsador 4
R6
1K
J1
C2
100nF
C1
10UF/16V
IC1
LCD 2x16 I2C
VCC
SDA
SDL
GND
CN2
Receptor
HFBR-2521
N.C.
Rx
R8
1K
C3
100nF
IN
SW?
CN1
Emisor
HFBR-1521
Tx
R10
1K
R9
220
Q2
BC817
Q3
BC817
OUT
J2
Rele Precarga
Comparador Shunt 1
Comparador Shunt 2
INH Pulsos
IN DIGITAL
S1
S2
Fallo Comparador SHUNT 1
Fallo Comparador SHUNT 2
AR?
Arduino Nano
VIN
5V
3.3V
RESET
PB0/CLK0
PB1
PB2/SS
PB3/MOSI
PB4/MISO
PB5/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/T0
PD5/T1
PD6/AIN0
PD7/AIN1
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
PC6/RESET
ADC6
ADC7
AREF
Reset Shunt 1
Reset Shunt 2
Pulsador 4
J4
PULSOS OUT
Pulsos A
Pulsos B
J5
S4
S6
J3
IN TENSIONES
Tension +12V
Pulsos Ventilador
Tension Bus de continua
Tension Sensor Temperatura
Tension Sensor Hall 1
Tension Sensor Hall 2
C5
100nF
C4
10UF/16V
R14
1k5
R16
1K
R13
1K
R15
1K
R2
1k
R3
1k
Q1
BC817
INH Pulsos
R11
1k
R12
1k
Q4
BC817
Rele Precarga
GND
ARDUINO
S6
S4
S2 S1
J2
J1
J3
J4
Hab. Shunt 2
Hab. Shunt 1
S1 Habilitación Shunt 1
S2 Habilitación Shunt 2
S4 Activar/desactivar Reset Shunt 1
S6 Activar/desactivar Reset Shunt 2
Tension Sensor Hall 2
Tension Sensor Hall 1
Tension Sensor Temperatura
Tension Bus Continua
Pulsos Ventilador
Tension +12V
Title
Size
A3
Number
Revision
Date:
26/11/2021
Sheet
of
File:
G:\Mi unidad\..\Arduino.SchDoc
Drawn By:


PR GBU1010
J1
CON IEC
F1
Fusible
TR1
T 1 X 12V 10VA
PR1
U1
L7812
Vin
GND
Vout
C1
10UF/35V
C3
100nF
C4
100nF
C2
100uF/25V
+9V
R1
1k
D1
ROJO
CN3
Emisor
HFBR-1521
Tx
N.C.
OUT TX
R4
220
Q1
BC817
R5
1K
Q2
BC817
CN4
Receptor
HFBR-2521
Rx
C5
100nF
R6
1K
IN RX
AR1
ARDUINO DUE
ICSP
POWER
COMMUNICATION
ANALOG IN L
ANALOG IN H
DIGITAL I/O
DIGITAL I/O PWM
DIGITAL I/O PWM
DIGITAL I/O
Arduino Due
J5
J3
J12
GND
GND_1
Pulsos 4 1
Pulsos 4 2
R2
220
R3
220
CN1
CN2
4 1
3 2